TECHNISCHE
UNIVERSITÄT
DARMSTADT

# SERVICE-ORIENTED COMPUTING *with* WIRELESS PARTICIPANTS – ADAPTATION MECHANISMS *and* DECISION SUPPORT *for* LIGHTWEIGHT WEB SERVICE CONSUMPTION

Vom Fachbereich Elektrotechnik und Informationstechnik
der Technischen Universität Darmstadt
zur Erlangung des akademischen Grades eines
Doktor-Ingenieurs (Dr.-Ing.)
genehmigte Dissertation

von

DIPL.-ING. APOSTOLOS PAPAGEORGIOU

Geboren am 4. Oktober 1983 in Athen, Griechenland

Vorsitz: Prof. Dr.-Ing. Hans Eveking
Referent: Prof. Dr.-Ing. Ralf Steinmetz
Korreferent: Prof. Dr. rer. nat. Claudia Linnhoff-Popien

Tag der Einreichung: 22. März 2012
Tag der Disputation: 24. Mai 2012

Hochschulkennziffer D17
Darmstadt 2012

## ABSTRACT

At the crossroads of two hot trends of modern computer science, namely service-orientation and mobile computing, great potentials arise together with tough challenges. The list of advantages of combining these technologies is long and compelling: Outsourcing of data- and processing-intensive software tasks from mobile devices to more capable systems, quick mobile application development through the use of existing software services that are otherwise difficult to implement on mobile devices from scratch, and the list goes on. However, service-oriented messaging and description technologies are characterized by a verbose, self-descriptive nature, in order to achieve interoperability and platform-independence. This renders them often heavyweight and thus not always a good match for the resource-constrained nature of mobile, wireless devices.

In the face of the fact that very different devices will coexist in future service-oriented systems, researchers are diligently trying to tackle the aforementioned challenges by developing adaptation mechanisms. These are meant to allow the systems to dynamically and seamlessly switch to configurations that suit each particular participant. When it comes to one of the most important service-oriented technologies, namely Web services, adaptation for resource-constrained devices usually translates to an attempt to use lightweight ways of communication that are dictated by the system context.

The striving for enhancements in the field of Web service adaptation mechanisms for wireless participants starts with three simple questions: Which specific adaptation mechanisms can be used and what are their characteristics? Can the development of new adaptation mechanisms contribute to the currently existing possibilities? Which adaptation mechanisms should be used under which wireless system contexts? The systematic and scientific examination of these questions leads to the three corresponding main contributions of this thesis.

First, the conducted survey and comparison of Web service adaptation mechanisms is the first work examining the conditions of the wireless system context under which the adaptation mechanisms achieve significant benefits. Thus, the survey results can be used as a basis for examining the two further issues. Second, a new, caching-based Web service adaptation mechanism is presented. The developed mechanism is the first to enable the use of cached responses of external, i.e., third-party, Web services with guaranteed 100% freshness in an automated and generic manner. The freshness of cached objects refers to their probability of being up-to-date. The evaluation proves that the approach can lead to performance enhancements of mobile Web service invocations compared to other approaches that achieve absolute freshness. Third, provided that no single adaptation mechanism is the best-performing under all possible system contexts, the thesis offers insights with regard to the issue of corresponding decision support. Decision support algorithms that are based on the obtained survey results are developed and evaluated with focus on an important, yet uninvestigated aspect, namely the existence of missing system context data. The

respective work reveals which data imputation approaches are best suited to the examined scenario.

By discussing the above issues within a well-defined future scenario of mediated Web service usage, the thesis provides further contributions, such as architectural solutions, engineering approaches, and problem formulations. All in all, its findings should be interesting for any work in the research area of mobile services.

# KURZFASSUNG

An der Schnittstelle zweier aktueller Trends in der Informationstechnik – Serviceorientierung und mobilem Computing – ergeben sich sowohl große Potentiale als auch Herausforderungen. Die Kombination dieser beiden Technologien bietet sowohl funktionale als auch nicht-funktionale Vorteile. Sie ermöglicht u. a. die Umschichtung daten- und verarbeitungsintensiver Software von mobilen auf performantere Hardware und die schnelle Entwicklung von Mobilanwendungen durch die Nutzung existierender Software Services, die nur schwer auf den mobilen Endgeräten implementiert werden können. Allerdings verwenden serviceorientierte Systeme zur Erreichung ihrer Ziele, darunter Interoperabilität und Plattformunabhängigkeit, in der Regel selbstbeschreibende und somit schwergewichtige Nachrichten- und Beschreibungsformate. Die Verwendung dieser Technologien auf mobilen, drahtlosen, in ihrer Leistung beschränkten Endgeräten, stellt sich somit als schwierig dar.

Zukünftig wird eine große Zahl heterogener Endgeräte in serviceorientierten Systemen Anwendung finden. Daher wird in der Forschung verstärkt die Entwicklung passender Adaptionsmechanismen betrachtet. Diese Mechanismen sollen es einem System erlauben, dynamisch und nahtlos zwischen verschiedenen Konfigurationen umzuschalten, welche auf die Anforderungen der jeweiligen Endgeräte zugeschnitten sind. Im Hinblick auf den wichtigsten Standard im Umfeld serviceorientierter Systeme, d. h. Web Services, besteht eine solche Adaption häufig in der Erreichung einer leichtgewichtigeren Kommunikation, die sich an den Vorgaben des aktuellen Systemkontexts orientiert.

Ausgangspunkt der Forschung zu Verbesserungen im Hinblick auf Web-Service-Adaptionsmechanismen für drahtlose Endgeräte sind drei grundlegende Fragen: Welche spezifischen Adaptionsmechanismen können verwendet werden und welche Charateristika besitzen diese? Kann die Entwicklung neuer Mechanismen weitere prinzipielle Möglichkeiten der Adaption eröffnen? Welcher Adaptionsmechanismus sollte in welchem mobilen/drahtlosen Systemkontext verwendet werden? Die systematische, wissenschaftliche Untersuchung dieser drei Fragestellungen führt zu den drei wesentlichen Beiträgen dieser Arbeit.

Erstens enthält diese Arbeit eine Erhebung und einen Vergleich verschiedener Adaptionsmechanismen. Erstmalig wird in diesem Zusammenhang der Nutzen aller Mechanismen unter gegebenen Kontexten mobiler und drahtloser Systeme untersucht. Diese Untersuchung bildet die Grundlage zur Betrachtung der weiteren Fragestellungen. Zweitens wird ein neuer, Caching-basierter Adaptionsmechanismus für Web Services vorgestellt. Der Mechanismus ermöglicht erstmals die Verwendung von im Cache zwischengespeicherten Antwortnachrichten für externe – d. h., durch Dritte bereitgestellte – Web Services bei gleichzeitiger Sicherstellung von 100% *Freshness* in einer automatisierten und generischen Weise. Die Freshness von zwischengespeicherten Objekten bezieht sich auf die Wahrscheinlichkeit, dass ihre Inhalte aktuell sind. Im Rahmen einer Evaluation wird die Möglichkeit von Performanzverbesserungen beim mobilen Aufruf von Web Services im Vergleich zu anderen Ansätzen, die die Freshness garantieren können, nachgewiesen. Drittens wird – beruhend auf

der Feststellung, dass kein Adaptionsmechanismus über alle Systemkontexte hinweg grundsätzlich die beste Performanz liefert – die Möglichkeit der Unterstützung bei der Auswahl von Adaptionsmechanismen untersucht. Entsprechende Entscheidungsalgorithmen werden auf der Grundlage einer vorhergehenden Erhebung entwickelt. In der hierauf aufbauenden Evaluation wird schwerpunktmäßig der bedeutsame, aber bislang unzureichend betrachtete Aspekt der Unvollständigkeit der den Systemkontext beschreibenden Daten untersucht. Aus den Ergebnissen ergeben sich Empfehlungen hinsichtlich der Wahl einer geeigneten Imputationsstrategie für das untersuchte Szenario.

Die Arbeit betrachtet ein wohldefiniertes, künftiges Szenario der Web Service-Nutzung über eine Vermittlungsschicht. In diesem Rahmen enthält die Arbeit weitere Beiträge, darunter Architektur- und Entwicklungsansätze sowie Problemformalisierungen. Die vorliegende Dissertation stellt somit einen wichtigen Beitrag für künftige Arbeiten im Umfeld der Forschung von mobilen Services dar.

# ACKNOWLEDGMENTS

First of all, I would like to thank Prof. Dr. Ralf Steinmetz for his supervision and for creating and leading KOM (Multimedia Communications Lab), an inspiring and perfectly organized lab, at which I had the opportunity to work and write my PhD thesis. I would also like to the thank Prof. Dr. Claudia Linnhoff-Popien for the second assessment of my thesis.

My special thanks goes to my colleagues of the "Service-Oriented Computing" research group of KOM, namely Dr. Nicolas Repp, Dr. Julian Eckert, Dr. Stefan Schulte, Dr. André Miede, Dr. Michael Niemann, Dieter Schuller, Sebastian Zöller, Ulrich Lampe, Melanie Siebenhaar, Ronny Hans, and Olga Wenge for the excellent cooperation, the useful advice, the permanent support, and the wonderful working environment. However, the rest of my colleagues at KOM deserve my gratitude as well, both the scientific staff for the knowledge exchange and the administrative staff for their continuous support.

Further, I would like to thank my students, especially Marius Schatke, Sebastian Ahlfeld, and Jeremias Blendin for contributing so much to my work, as well as all members of the scientific community who constantly offered useful feedback through conferences and project meetings. A special thanks goes to my project partners, especially Anke Thede and Bastian Leferink, because a big part of the success of my work and our projects is owed to them.

Finally, I want to say "thank you" to the people who give meaning to everything I do, namely my family and my friends, but also to all the people that accompanied me during this "journey" of the last four years, making my life happy and my purpose meaningful. However, I will leave my parents out of all this, because saying "thanks" to them is not enough and no words exist that could express the importance of their love and their support.

*Darmstadt 2012* Apostolos

# CONTENTS

# 1

# INTRODUCTION

»Well begun is half done.«

— Aristotle

The use of wireless (handheld, embedded, and mobile) devices is increasing [64, 107] in a scale that inspires people to argue that we are finally entering the *post-PC era*, which had already been proclaimed as early as 1999 [118]. Smartphones and other cellular devices, Personal Digital Assistants (PDA), vehicular systems, and sensor motes are examples of such devices. The different wireless access networks, protocols, and standards that are used to interconnect them are gaining in importance, as well. However, these devices, which are usually limited in size and capabilities compared to server systems or stationary Personal Computers (PC), do not comprise the only domain that is racing forward. Highly capable systems, enabled by the never-ending development of hardware, but also by grid and cloud computing [9, 151], are not likely at all to play a less important role in the future [9]. As usual, the truth is somewhere in between: Devices and connections of different types and capabilities will most probably coexist in future systems.

This coexistence requires the utilized design paradigms and computing approaches to be flexible and technology-agnostic. For systems where interoperability, platform-independence, and extensibility are considered important, *service-orientation* is very likely to be a dominating aspect [35, 106]. According to the paradigm of Service-Oriented Computing (SOC), software is built in the form of autonomous services that can be published, discovered, executed, and composed platform-independently [35, 57, 106], based on certain technologies and standards for their description, discovery, and messaging. The corresponding architectural paradigm is called Service-Oriented Architecture (SOA), while the use of services can cross the boundaries of design and architectural paradigms, as is the case in the evolving Internet of Services (IoS) scenario [22, 136]. In the IoS, services are seen as (tradable) goods and a high number of them are offered by different providers to diverse clients through global service marketplaces.

When service-orientation is adopted in the field of mobile and wireless computing, both great *potentials* and tough *challenges* appear. On the one hand, the reuse of logic that is difficult to implement on the devices from scratch[1], the flexible outsourcing of data- and/or processing-intensive software in the form of external services, and the convergence of mobile applications with enterprise systems which are already built according to the SOC paradigm [78], are all very appealing. On the other hand, SOC technologies are based on the principle of self-description in order to achieve

1 The mentioned implementation difficulty might have diverse causes. Mobile programming languages often offer only a subset of the commands and the capabilities of traditional programming languages, they are sometimes characterized by a lack of libraries, while the installation and usage of database management systems is much more constrained and challenging. Finally, even if such obstacles can be overcome, why should logic that already exists be re-implemented with a different technology?

interoperability [57]. This means that the communication may include considerable overhead compared to traditional ways of network-based communication [157], e.g., for the processing and exchange of extra data and metadata. Although powerful enterprise systems have rarely been affected by this, it may cause performance degradation and higher costs for wireless devices [26, 160] or embedded systems [45]. Indeed, services are often designed target system-agnostically, without being tailored to the features of particular clients. A discussion dedicated to the issues of *mobile services computing* can be found in [178].

A drift away from the SOC paradigm when devices are likely to be involved is not an option. This is both because SOC matches the needs of particular application types (mobile payment, mobile location-based services etc. [115]) and because the paradigm has further advantages for mobile computing (cf. the previous paragraph). Thus, if it is desired to vanquish the drawbacks while retaining the advantages, SOC-related technologies should be seamlessly and efficiently adaptable to the needs of different system participants.

## 1.1 MOTIVATION

Although enterprise systems have been the originating and traditional domain of SOC, "everyday" applications have also begun to play a very important role in this field. A recent survey of TechTarget [40] positioned Web service-based mobile apps at the second place in the category "service-based implementations planned for the future" (planned by 60% of the questioned developers/companies), even higher than the composite application assembly (planned by 58%), which has been often named as SOC's main potential [105].

Although some argue that the constraints of mobile devices (limited bandwidth, CPU, memory, or energy resources) are disappearing due to technological progress, the gap will not cease to exist. This is indicated by the latest analyses of future wireless communications. For example, in the book of Sesia et al. [142], five categories of user equipment are defined, with smartphones being placed only in the second or third category. According to this categorization, devices of higher categories will be able to use wireless internet connection rates up to six times greater than those of lower categories. Furthermore, the wired connections of the future will be even faster than that, not to mention the fact that devices less capable than smartphones, such as sensor nodes, will be able to participate in service-oriented systems. Thus, the big differences in device capabilities and connection qualities will maintain the need for *adaptation*, also because the amounts of the data that are processed and wirelessly transmitted are growing parallel to all other technological developments [17].

Most efforts for adapting SOC-related technologies to the needs of different, resource-constrained participants focus on techniques for reducing the amount of data that is processed and wirelessly transmitted during the usage of XML-based Web services in order to enhance performance. The latter, from now on simply referred to as Web services (WS), constitute the most common technology for SOC implementations. This is also the focus of the work-at-hand.

A look at the timeline of related research activities (cf. the surveys in [100] and [157]) reveals that the aforementioned efforts towards scenario-dependent, lightweight (i.e., less data- and processing-intensive) Web service communication began with intuitive

solutions such as device- and connection-dependent compression of Web service messages [160, 168]. Approaches for seamless transformation of the communication protocol or format followed [2, 3, 7], while the most recent efforts are inspired by more specialized features of Web service communication, such as the involvement of Web services in known, repeatedly executed processes (so that prefetching of information can be a solution), or the fact that Web service messaging, especially multicasting, includes redundancies that can be eliminated [97, 135, 156]. In parallel to those works, other solutions, such as caching-based [79] or transport-protocol-based [116] solutions, have appeared.

Some of these approaches can be easily applied in the form of adaptation mechanisms for existing Web services, while others would require re-designing protocols or re-building system parts from scratch. Nevertheless, they are all important on the way towards efficient and seamless adaptation of Web service communication. What is missing comprises the offering of generic and automated Web service adaptation mechanisms that correspond with a particular class of data reduction techniques, namely those that directly handle the actual content and not just the format or the representation of the messages. Furthermore, this development should come along with an underlying intelligence that supports efficient selection and combination of the possible adaptation mechanisms based on the context and the characteristics of modern systems.

## 1.2 RESEARCH GOALS AND CHALLENGES

The identification of the aforementioned gap in related work, along with the further examination of issues that have arisen during the development and operation of a Web service adaptation layer for wireless participants of the IoS [99], drew the focus of the thesis to the pursuit of the following main research goals:

- Examination and categorization of the most important Web service adaptation mechanisms, together with an analysis that reveals their requirements and limitations when they are applied in order to support wireless participants.
- Design and development of a new adaptation mechanism which complements existing solutions by addressing limitations in the area of generic and automated mechanisms for reducing the content of Web service messages.
- Development and examination of a decision support system that indicates which adaptation mechanism(s) should be enabled for particular Web services based on the system context.

All three described goals are components of the more general objective of *improving the performance of Web service usage on wireless and/or limited devices*. Indeed, the focus is on *usage*. The examination of the *hosting* of Web services on limited devices is *not* among the goals of the thesis. Further, concerning the development of a new mechanism, the goal is *not* to develop an adaptation mechanism that is universally "better than the others", but an adaptation mechanism which creates new potentials and would be preferred over the others under certain circumstances.

Among many challenges that appear while pursuing the goals described above, the most important challenges from a scientific point of view are the following:

- It is hard to adapt Web service technologies without loss of interoperability, as it is exactly the same feature of service-orientation that is responsible for both interoperability and performance limitations, namely *self-description* [57].
- In the IoS, which is the target scenario of this thesis, users (or mediators) normally do *not* have access to the service implementation or to the provider's system. Thus, some approaches for lightweight Web service communication, such as [154], cannot be implemented as adaptation mechanisms of existing third-party Web services, because they would require many changes at the side of the service provider.
- Particularly for approaches that adapt the content of Web service messages, generic solutions are very difficult to implement, because there is an infinite number of possibilities concerning the way this content may look (e.g., included information, used data types).
- Approaches for lightweight Web service communication have usually been developed in different contexts and often with different motivating scenarios [157]. Therefore, they have been evaluated from different perspectives, so that comparing them is not as easy as putting a couple of experimental results next to each other. Instead, their assumptions about the mobile computing context and the conditions under which they perform better than standard communication schemes have to be analyzed.

## 1.3 CONTRIBUTIONS

Although splitting the overall goal into more concrete research goals is a very important step, a quest for solutions to the most important challenges is a very lengthy process, which does not necessarily lead to significant contributions, at least not without innovative ideas and proper design decisions. In the following, the main contributions of the thesis are listed. Although there is a direct (1-to-1) association between the contributions and the previously stated goals, the contributions refer to the concrete outcome of the work. As different decisions would have resulted in a different outcome, the description of the contributions is followed by a very brief summary of important decisions, characteristics, and limitations of the approach.

- The first contribution is a survey that analyzes, categorizes and formally compares many of the most important published approaches for (performance-oriented) Web service adaptation. To the best of my knowledge, it is the first survey that analyzes the conditions of wireless communication (network-, device-, service-, and application-features) under which each of the mechanisms can be considered to be appropriate. The survey complements more general (i.e., not oriented to wireless communication and not focused on adaptation) surveys of approaches for Web service performance enhancements, such as [157].
- The core contribution of this thesis is the development of a new, caching-based Web service adaptation mechanism, which remedies a well-known restriction of Web service caching in an automated and generic manner. This restriction is implicitly indicated in [159] and [152]. To the best of my knowledge, it is

the first approach that automatically allows for the use of cached Web service responses guaranteeing 100% *freshness* generically, i.e., for any Web service, even if the latter is provided by a third-party. The freshness of cached objects refers to their probability of being up-to-date.

- A further contribution is the development and examination of decision support for Web service adaptation. Accordingly, scoring algorithms have been developed, based on the results of the related survey (cf. first contribution). The previously mentioned algorithms score the adequacy of different Web service adaptation mechanisms under changing conditions. As the scoring algorithms often have to work with *missing data*, the best means of tackling this problem are evaluated. The issue of missing data concerns the handling of unknown values that appear for monitored system context attributes and is an important issue, which has not been directly addressed for this scenario before.

The results of the survey are based on the information included in related publications, but also on complementary experiments. However, experiments have been used only to validate incompletely proven statements or implicit allegations of the related work and not in order to draw new exclusive conclusions of my own. Another important decision was the use of discrete categorical (and not numerical) data. Otherwise, the high number of unknown exact values and the related uncertainty factor would make it difficult to characterize the approaches. Thus, the results are not appropriate for reaching absolute statements of the type "A is better than B", but they achieve the goal of creating a basis for decisions.

The core innovative ideas of the newly developed caching-based adaptation mechanism comprise the automated Web service proxy generation and the generic enablement of validity checks for Web service responses. In the designed approach, a proxy-based solution has been preferred over a solution that would require changes at the server-side. Direct comparison of contents has been preferred over comparison of XML documents. Further, automated software generation has been preferred over a priori development and deployment. Finally, absolute freshness of the used information has been set as a requirement (which is rarely achieved by caching approaches). As a result, the approach has the limitation of the inability to spare connection establishments, but it can achieve significant performance enhancements, while being unique in that the performance enhancements come with a freshness guarantee. Another limitation is that the algorithms used (and the evaluation results) are bound up to a certain extent to characteristics of modern Web service technologies. However, firstly, Web services are here to stay and, secondly, the core ideas and many algorithmic parts could be transferred to different –but similar– technologies.

Last, the developed scoring algorithms have not been evaluated and compared based on real scenarios because the sets of correct decisions defined for any scenario would be subjective. Instead, the evaluation focuses on how the algorithms behave in the presence of missing data, with missingness scenarios that reflect the examined case. Thus, the results may not prove the superiority of any scoring approach, but they can offer new insights that are interesting to decision support system designers by proving the superiority of particular approaches for handling scenario-related missing data.

## 1.4 THESIS ORGANIZATION

The remaining chapters of the thesis are structured as follows:

Chapter 2 provides the necessary background by describing core concepts and technologies of the scientific fields in which the thesis is positioned, namely SOC and mobile computing. Furthermore, an overview of other related subjects, such as Quality of Service (QoS) and caching, is given.

Chapter 3 introduces adaptation as a general, interdisciplinary approach for enhancing systems performance. Then, it provides the transition from adaptation in general to adaptation in SOC, and, finally, to the adaptation of Web services for wireless participants, which is the focus of the thesis.

Chapter 4 explores the subject of Web service adaptation for wireless participants by concretizing the framework and the scenario of the conducted research and by presenting the results of a survey of Web service adaptation mechanisms. Although the examined adaptation mechanisms cover an important part of the related approaches, more specialized related work about "Web service response caching" and "decision support" is provided in Chapters 5 and 6, respectively.

Chapter 5 describes the conception, the design, the most important technical and algorithmic details, and a thorough evaluation of the first solution for generically and automatically enriching the communication with any third-party Web service in a way that cached responses can be exploited while a freshness of 100% is maintained.

Chapter 6 describes and evaluates algorithms developed for the decision support of systems that include adaptation mechanisms such as the ones described in Chapters 4 and 5.

Chapter 7 summarizes the content, the main contributions, and the lessons learned throughout this thesis. Finally, it discusses how future research can build upon the foundations or extend the results of the work at hand.

# 2

# BACKGROUND

»The beautiful rests on the foundations of the necessary.«

— Ralph Waldo Emerson

AVENTURE that is concerned with the adaptation of Web services for mobile wireless usage belongs to the intersection of the scientific domains of Service-Oriented Computing (SOC) and Mobile Computing. This chapter provides a background for understanding the rest of the thesis by introducing these scientific domains with emphasis on the aspects and the technologies that are most relevant for the subsequent chapters. Further fundamentals that are important to the thesis, namely Quality of Service (QoS), Quality of Experience (QoE), caching, decision support, and missing data, are also briefly explained.

## 2.1 SERVICE-ORIENTED COMPUTING

SOC is a computing paradigm according to which software engineering is based on autonomous, loosely-coupled, and platform-independent networked services, which can be discovered and invoked by different participants and for different applications [106, 167]. By designing each service in such a way that it corresponds with a well-defined programming logic or business functionality and by managing, combining, and organizing these services inside Service-Oriented Architectures (SOA), the goal of SOC is to achieve higher levels of system flexibility, organizational agility, and software reusability [35, 57, 106]. Due to these features, SOC is even being discussed as the paradigm for the future internet architecture [89]. In order to achieve the mentioned service characteristics and desired system advantages, the SOC paradigm is framed by a set of basic concepts and principles and is realized by relying on technologies that adhere to these principles, thus enabling service-oriented implementations.

Together with *Grid*, *Peer-to-Peer*, and *Cloud Computing*, SOC comprises one of the most prominent paradigms for realizing distributed computing systems:

- While Grid Computing is based on the principle that the participants provide their resources [151], in SOC the participants only offer services with well-defined interfaces. Simplified, this difference can be translated into the "provision of hardware (processing power and memory) for the needs of remote applications" vs. the "provision of software (service implementations) for the needs of remote applications".
- Although such a difference does not exist in this form when comparing SOC with Peer-to-Peer, the latter paradigm is based on slightly different principles and focuses on different goals. As can be read in [150], Peer-to-Peer dictates the equality of the partners and focuses on decentralization. On the other hand, SOC focuses on interoperability (which in Peer-to-Peer is usually *assumed*

because of the partners' equality) and retains many characteristics of centralized approaches, e.g., the existence of service consumers and service providers in accordance with the roles of the client/server paradigm. Practical experience with SOC and Peer-to-Peer implementations also shows that peers of a single Peer-to-Peer system are normally running very similar (if not exactly the same) software, while this is not at all the case for the participants of a service-oriented system. Some more specific differences between SOC and Peer-to-Peer, also explained in [150], can be summarized in the following: (i) SOC implicates the use of machine-readable but verbose technologies in order to achieve interoperability, while Peer-to-Peer is not based on interoperability standards, but it rather assumes that the peers "understand" each other, (ii) SOC usually relies on centralized mechanisms for the publishing and the discovery of services, while in Peer-to-Peer systems discovery is based on distributed algorithms, and (iii) SOC relies on standard Internet Protocol (IP) addressing with the use of the Domain Name System (DNS), while Peer-to-Peer assumes that the peers might not have a permanent IP address and perform a distributed, overlay-based addressing scheme.

- Finally, the Cloud involves the use of many concepts and technologies of SOC and it rather complements SOC in that it provides a "value-added underpinning for SOC efforts" [120] by helping outsource the Information Technology (IT) resources (memory, computing, administration tasks, and more) to big server clusters and data centers, though not harming the SOC principle of building applications based on standardized, interoperable and reusable components. However, new dimensions appear for the problems of service provisioning and service distribution in the Cloud, so that new corresponding optimization approaches are being developed [71, 145].

Before the exploration of the concepts, the principles, and the technologies of the SOC paradigm, a brief explanation of the idea that lies in the core of the paradigm is provided. This core idea is usually explained based on a simple role model that defines three main roles (service provider, service consumer, and service broker) and their relationships and interactions. The latter are represented as a triangle where the service provider publishes a service that is discovered by the service consumer over a service broker, so that the service consumer and the service provider can then interact directly. This representation, namely the SOC-triangle (or SOA-triangle), can be found in most of the related literature. In Figure 1, this triangle is used as the basis upon which the main needs for standards and technologies in the SOC domain are illustrated. More concretely, the fact that service providers have to publish their services to service brokers or registries raises the need for standards and technologies for the description of services, such as Web Service Description Language (WSDL)[1] [166] or Unified Service Description Language (USDL) [22]. Through such descriptions the consumers can find out where the running services can be found and how they can be addressed or used in a platform-independent way, i.e., avoiding that the consumers have to care about the technologies utilized for the service implementation. Similarly, standards and technologies are needed for the discovery and the messaging (cf. Figure 1).

1 `http://www.w3.org/standards/techs/wsdl` (Last accessed in January 2012).

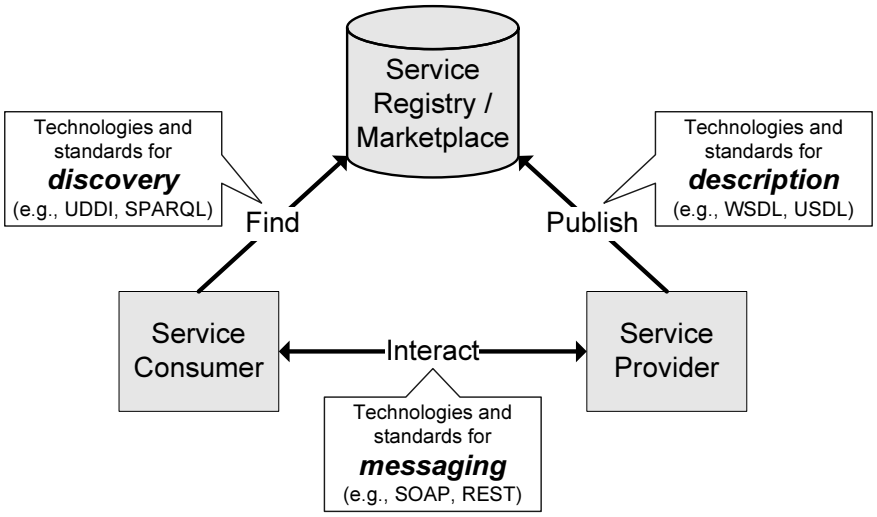



Figure 1: Basic model of SOC roles, interactions, and aspects

The SOC-triangle illustrates only a core idea, which has been complemented by a series of concepts and principles that evolved and matured during the application of the paradigm in the recent years. It is remarkable, for example, how strongly many researchers have associated SOC with service workflows and service composition (more about them is explained in the following), because of the interesting research questions that appeared in that direction. However, service workflow management and service composition are specific issues, which do not comprise the core idea of SOC (cf. Figure 1), but arise as a consequence of the broad application of this core idea and the existence of many well-described and interoperable services. The main concepts, principles, and characteristics that complement this core "triangle"-concept are described in the following.

### 2.1.1 *Basic Principles*

Only by adhering to certain principles can service-oriented systems achieve the advantages mentioned in Section 2.1. For example, the advantage of *system flexibility* is only achieved when the service design follows the principle of *loose-coupling*, because only loosely-coupled services can be easily exchanged or enhanced without affecting the rest of the system. With the goals of system flexibility, organizational agility, and software reusability in mind, one can learn from the related literature [35, 57, 106, 166] which principles are considered to be the most critical towards achieving these goals. The most important of these principles (based mainly on [57] and [35]) are summarized here, in order to discuss what they imply for the used technologies.

SERVICE AUTONOMY AND LOGIC ENCAPSULATION refers to the fact that the tasks that can be performed by a service do not depend on any other specific software implementations in order to have a meaning and be usable and valuable. The internal implementation of this autonomous service logic is hidden (encapsulated)

but the logic itself can be nevertheless understood and exploited by potential consumers.

INTEROPERABILITY AND PLATFORM-INDEPENDENCE refer to the fact that the service participants (providers and consumers) can be heterogeneous in terms of hosting environment (device, operating system etc.), programming language, application characteristics, and more. The knowledge of well-defined interfaces is enough, so that the underlying technologies may remain hidden.

LOOSE COUPLING refers to the fact that the services and their interactions are designed and performed in a way that the modification, or even the retirement, of a service does not render the systems that use them (or any other service) useless. Instead, services can be modified or exchanged with small or no impact.

The adherence to these principles can never be guaranteed alone by the existence of certain technologies, but it rather depends on an appropriate design and development, as well. However, in order to have the tools for such an attempt at the first place, enabling technologies are necessary. Thus, the following question arises: *What are the characteristics that software technologies must have in order to support the realization of these principles?* With retrospect to the technologies that have prevailed as main SOC enablers, with a careful examination of the standards that have been established, but also based on similar remarks of many researchers, e.g., [57, 110], it can be argued that the common characteristics of SOC-enabling technologies, which help adhere to the aforementioned principles, are the following:

SELF-DESCRIPTIVE: As platform-specific formats, specialized encodings, and proprietary methods for client-server communication would harm the autonomy and the interoperability, SOC technologies are not limited to providing or exchanging data, but they must almost always accompany the data with metadata (= data used to explain, describe, or annotate other data[2] [169]). Thus, both the descriptions of the services and the messages that they exchange are self-describing, i.e., they include metadata that help service participants of all types and platforms understand them.

WEB-DRIVEN: Interoperability means among others that communication may often span the borders of enterprise networks, while loose-coupling means that the providers of particular services may change after the system design phase, which, in turn, means that service communication may run over different networks at any given time. For this, SOC technologies cannot be designed with a logic that matches, for example, local or proprietary networks. Instead, SOC technologies are driven by the features and the trends of the Internet.

These two characteristics are of special importance to this thesis, because they are responsible not only for the main advantages but also for some major drawbacks of SOC technologies, which will have to be handled. As already mentioned, the advantages are mostly related to interoperability and platform-independence, while the disadvantages are related to verbosity and performance.

Other concepts and principles of SOC exist as well, but are not elaborated here because of their small relevance to the goal of the thesis. However, they have to be

2 `http://www.techterms.com/definition/metadata` (Last accessed in January 2012).

mentioned for the sake of completeness. These concepts include service composability (into workflows or processes), service discoverability, service statelessness, interface abstraction, business logic-driven granularity, and more [35, 66]. Accordingly, intensively researched issues that are inspired by these concepts and principles are related to the problems of service selection (for composed workflows), service resource planning, workflow similarity calculation, and SOA Governance. [13], [137], and [138] present heuristic solutions for the service selection and composition problem, [33] presents a heuristic solution for the service resource planning problem, [32] discusses the impact of pricing models on the service selection problem, [172] discusses approaches for calculating workflow similarity, while [93] proposes a generic framework for SOA Governance approaches.

### 2.1.2 *Technologies*

On the way from principles and characteristics down to technologies, it becomes clear that the self-descriptive and the web-driven nature that SOC technologies should have has played a very important role in the evolution of standards and in the selection of protocols (or other basic technologies). Almost all established standards are based on XML (eXtensible Markup Language) and most of the service-oriented communication runs over HTTP (Hypertext Transfer Protocol), while IP (Internet Protocol) is the most commonly used addressing protocol. XML is a flexible text format with a set of encoding rules, designed to carry self-descriptive data[3], while HTTP is the standard protocol for sharing information over the Internet[4]. Indeed, SOC literature, e.g., [35, 57, 166], almost always highlights XML and HTTP as the two basic standards upon which most of the SOC-specific standards have been built. At the transport layer, Transmission Control Protocol (TCP) is normally used, although User Datagram Protocol (UDP) is theoretically also a possibility.

A particular collection of SOC-specific standards that have been defined, among others, for the description, messaging and discovery of services are referred to as *standard Web services*, or simply *Web services* [57, 166]. Web services have become so popular as implementation means of SOC that the terms SOC and Web service are sometimes being confused or are thought to be inseparable. However, further standards and technologies that do not belong to this collection exist or may be developed. Three categories of standards can be derived directly from the basic SOC model (cf. Figure 1), namely for description, discovery, and messaging. Other standards that became popular were designed for the composition and choreography, but also for the security and reliability of Web services. The most popular standards are listed in the following, grouped by the purpose they are used for:

- WSDL is by far the most popular *description* standard, although further standards have appeared. The latter may focus on business and operational aspects, such as USDL, extensions of WSDL to support semantics, such as Semantic Annotations for WSDL (SAWSDL), or slightly different communication styles, such as Web Application Description Language (WADL).

3 `http://www.w3.org/XML`, `http://www.w3schools.com/xml/xml_whatis.asp`, `http://en.wikipedia.org/wiki/XML` (Last accessed in January 2012).

4 `http://www.w3.org/Protocols`, `http://en.wikipedia.org/wiki/Http` (Last accessed in January 2012).

- The Universal Description, Discovery and Integration (UDDI) standard has been specially designed in order to support *discovery* of Web services. However, other technologies, not directly related to the Web services standards collection, may considered to be important. For instance, Electronic Business using eXtensible Markup Language (ebXML) is used for defining and registering business processes, while SPARQL Protocol and RDF Query Language (SPARQL) is a related query language.

- For *messaging*, Simple Object Access Protocol (SOAP) is the traditional standard, although Representational State Transfer (REST) is also becoming popular as an alternative that achieves more lightweight communication by sacrificing transport protocol-independence and some possible interaction patterns [110].

- Business Process Execution Language (BPEL) has also become very popular as a standard for *composition and orchestration* of services, while ebXML can be considered relevant also for orchestration. The standard that is commonly used in order to graphically represent BPEL processes is called Business Process Model and Notation (BPMN).

- A big set of standards such as WS-security, WS-reliability, and Security Assertion Markup Language (SAML), but also basic XML-related techniques, such as XML encryption, frame the *security and reliability* of Web services.

Although many of the most popular standards and technologies have been mentioned above, other categories may also be identified. An overview of Web service standards can be found in [13]. Among them, WSDL and SOAP are the most relevant to the following chapters of this thesis. In fact, the combined use of WSDL and SOAP comprises the Web service realization of the Remote Procedure Call (RPC) concept [132, 155]. Therefore, the structures of a WSDL description document and of a SOAP message are explained in more detail in Figure 2. Although some fields are omitted, Figure 2 clarifies the main parts and the logic of the two standards. Knowing that a service can be used in order to perform particular actions (or tasks), which are called *operations*, the rest of the information provided in Figure 2 should be self-explanatory. The machine-readable formats of the documents do not only enable interoperability during the use of the service operations on different platforms, but they open the way for a variety of other actions, e.g., the automated semantic matchmaking of service descriptions in order to identify functionally equal or similar services [139], probably by using technology-specific query languages [140], or the automated identification of service changes (just by parsing the WSDL), in order to easily (or even automatically) adjust the service consumer applications accordingly [38].

Standards and technologies must, of course, come along with their implementations, as well as a variety of enablers: code libraries, tools, accompanying software that supports the deployment of services, service address resolution modules etc. These enablers are usually delivered packaged in middleware and/or software solutions, which are called *service platforms*. A service platform can vary from a simple enabling infrastructure (e.g., a single service registry with a SOAP engine) to a complex Enterprise Service Bus (ESB) [72]. More details about service platforms and the tasks that they undertake are provided in Chapter 3.

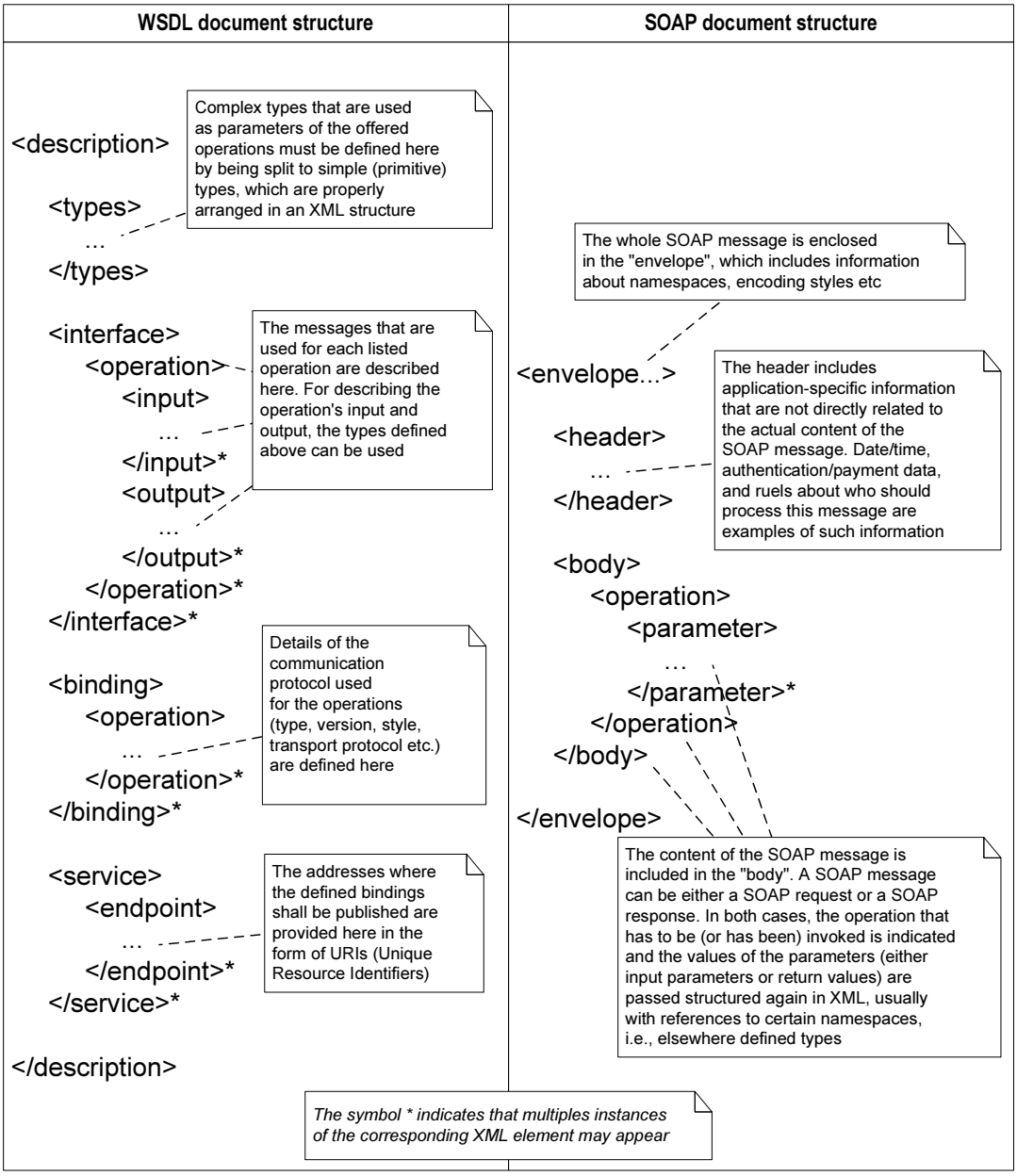



Figure 2: High-level document structure of the most important standards for WS description (WSDL, version 2.0) and messaging (SOAP, all versions)

## 2.2 MOBILE COMPUTING

As already explained in Chapter 1, mobile computing is becoming a very important domain for SOC and Web services. Commonly used definitions[5] [94], popular online sources[6], and scientific communities[7] "agree" that the term *mobile computing* is not limited to "mobility" or "movement", but has grown to include all aspects of "not physically connected" use of technology. Thus, the evolution of the wireless devices

5 `http://research.itknowledgehub.com/technology/wireless/mobile_computing` (Last accessed in January 2012).

6 `http://en.wikipedia.org/wiki/Mobile_computing` (Last accessed in January 2012).

7 `http://www.sigmobile.org/pubs/mc2r` (Last accessed in January 2012).

that can perform mobile computing and of the wireless networks that are involved in it are two central aspects of mobile computing.

Mobile computing is subject to certain limitations compared to server or desktop computing with fixed networks. These limitations, such as lower bandwidth, higher latency, less Central Processing Unit (CPU) power, and power consumption (limited battery), have already been identified as early as 1994 [39] and are undoubtedly still valid today and probably also in the future[6] [17, 135, 163]. [107] even mentions "automatic offloading of data traffic from mobile to fixed networks" as a future technique.

The reasons for the above limitations are in most of the cases obvious. CPU power and memory limitations are usually related to the fact that mobile devices are desired to be much smaller than stationary devices, because they have to be practical and discrete, while battery limitations stem from the fact that the devices cannot have continuous power supply from a network, because they have to be portable. A little more complicated are the reasons for the bandwidth limitations, i.e., the reasons why wireless communication is generally slower and more complicated than wired communication. The most important reason are the much bigger interferences from physical obstacles during the wave propagation, the electrical noise from the air, the fact that the "competition" for wireless frequencies among different parties is much more constrained and thus much harder than the competition for access to wired media, the constraints of different transmission antenna technologies and the fraction of the signal that is not received by the antennas, and more. The connection qualities change continuously even for the same connections, as the so-called signal strength is changing because of the above reasons. The notion of signal strength can, of course, be used also in wired networks but it is meanwhile trivial for them.

With the motive of examining and potentially enhancing the use of Web services in wireless environments, the types and the characteristics of modern *wireless devices* and modern *wireless access networks* are examined in the following subsections. The diversities and the limitations of mobile computing strengthen the motivation of this work, while the knowledge of its characteristics is necessary for developing approaches that suite it.

### 2.2.1 *Wireless Devices*

Since the appearance of Alexander Graham Bell's photophone in 1880 and the works of Guglielmo Marconi for long distance radio transmission in 1895, various types of wireless devices have appeared, many of which are still of importance for the scientific domain of wireless communications. However, the achievements that supported the transition from simple wireless communication (usually analogous, "voice-only") to modern mobile computing have been a product of the last two decades. These achievements were, firstly, the fast wireless transfer of digital data, and, secondly, the integration of enhanced processing capabilities in wireless devices. Furthermore, as also indicated by the survey of [107], almost 100% of the global mobile data traffic is produced by new generation devices, such as sensors, mobile phones, smartphones, laptops etc. Thus, only these types of wireless devices are relevant to Web service usage.

Table 1: Summary of recent classifications of wireless devices

| SOURCE | FOCUS | MAIN CRITERIA | IDENTIFIED CATEGORIES |
|---|---|---|---|
| Smura et al. [147] | Mobile service usage | Device size, circuit-switched call capability, operating system type | Mobile phones, smartphones/PDAs, ultra-mobile PCs, laptops and tablet PCs, other devices |
| Sesia et al. [142] | Cellular networks evolution | Max. data rates, receive antennas, modulation support, memory requirement for physical layer processing | User Equipment 1 - User Equipment 5 (The categories have no names; they refer directly to numerical values of specific characterictics) |
| Cisco Systems, Inc. [107] | Future mobile data traffic | Intuitive criteria, not explicitly listed | M2M (machine to machine), non-smartphones, smartphones, tablets, home gateways, laptops and netbooks, other portable devices |

Although different classifications of wireless devices can be found in literature, e.g., in [107, 142, 147], none of them indicates that the characteristics of the different devices are converging. [107] does not only predict that devices of different capabilities will coexist, but also that all the examined device types will have significant contributions to the global mobile data traffic in the future. Table 1 summarizes some recent classifications of wireless devices based on the current situation, but with an orientation to future technologies or forecasts about the future evolution of mobile computing. It must be mentioned that the examined devices are –or will very soon become– Web-service enabled.

In order to provide a feeling about the diversity of wireless device capabilities, the following *example* comparisons can be mentioned: The CPU power of a sensor mote (which would fall under the device category M2M of [107] in Table 1) may be up to two orders of magnitude smaller than that of a contemporary smartphone (cf., for example, [63]), while a smartphone is in turn many times weaker than laptops, PCs, and servers, has a battery lifetime of up to a couple of "hours of data access" [112], and would fall under the category "User Equipment 2" of [142] (cf. Table 1), which means that, in future cellular networks, other wireless devices will be able to use (cellular) connections that are up to six times faster [142].

Some current devices are particularly popular and representative of their categories. Such devices are also commonly used in experiments and evaluations that focus on the respective categories. For example, the iPhone of Apple, the Android-based smartphones of Google and Motorola, the E– and N–series of Nokia etc. are popular smartphones, while TelosB and Mica of Crossbow are commonly used sensor motes.

### 2.2.2 *Wireless Access Networks*

The term *wireless access networks* is normally used to characterize the wireless communication technologies that are used in order to offer to a wireless device access to a fixed (or core) network, usually the Internet. Although there are many different wireless communication technologies, not all of them are highly relevant to mobile computing. For example, *satellite networks* are used for other purposes, such as navigation, but they are not (yet) well suited for data exchange in mobile computing, mainly because of their technical complexity and their costs [125]. Another wireless communication technology that is not used in mobile computing is *broadcasting*, which is only suitable for radio and television, because it only supports one-way communication. The wireless communication technologies that should be considered in the context of mobile computing are mainly Wireless Local Area Networks (WLAN) and cellular networks, while Wireless Personal Area Networks (WPAN) may also play a role. It is the different variations of these three technologies that normally serve as wireless access networks for mobile computing devices.

WIRELESS LOCAL AREA NETWORKS are relatively fast but they usually have a quite short range. They are mostly based on the 802.11 family of standards of the Institute of Electrical and Electronics Engineers (IEEE). In the context of mobile computing and mobile Web service usage, they are important because wireless devices often use this technology in order to connect to an Internet access point, which grants them faster and cheaper access to the Internet compared to cellular networks. When using the standard frequency band of 2.4 Gigahertz (GHz), the latest WLAN versions (802.11n) currently have a maximum data rate of 72.2 Megabits per second (Mbit/s). However, users normally experience much lower rates for single connections, depending on the access point (router) technology, the load, and the signal strength. Furthermore, the vision of having so many WLAN access points that all users could be always connected over public or third-party access points [49] has not yet become a reality, not even in dense urban areas. Even if there is a possibility for this vision to overcome problems related to costs, technology, security, and privacy, WLAN-usage is at the moment very far behind cellular network usage when it comes to guaranteeing the "always-connected" feature of moving devices.

CELLULAR NETWORKS are used for mobile telephony and include different technologies for connecting to data networks, e.g., the Internet. Depending on the way (used frequencies, signal multiplexing techniques etc.) in which the devices communicate wirelessly with the base stations, but also on the way in which the core ("backbone") network operates, there are different possibilities for data access in a cellular network. The most widely used are the following: General Radio Packet Service (GPRS), Enhanced Data Rates for GSM Evolution (EDGE), High-Speed Packet Access (HSPA), and Universal Mobile Telecommunications System (UMTS), while Long Term Evolution of 3G mobile networks (LTE) is still not widely used but is considered to be the future standard in the domain [142]. The provided data transfer rates change continuously. GPRS and EDGE offer transfer rates of less than 0.5 Mbit/s, UMTS and HSPA can theoretically offer up to some tens of Mbit/s, while LTE has a theoretical maximum of approximately 100 Mbit/s (or even up to more than 300 Mbit/s when multiple

Table 2: Realistic characteristics of cellular networks according to [109] and [6]

| TECHNOLOGY | MIN. DELAY | MAX. DELAY | MAX. RATE |
|---|---|---|---|
| GPRS | 150 ms | 550 ms | 80 kbit/s |
| EDGE | 80 ms | 400 ms | 236 kbit/s |
| UMTS | 35 ms | 200 ms | 1920 kbit/s |
| HSPA | 35 ms | 200 ms | 3072 kbit/s |
| LTE | 12 ms | 50 ms | 7680 kbit/s |

antennas are used). Nevertheless, the realistic values that end-users experience are usually much lower than these. Table 2 shows the *realistic* characteristics of cellular network data access technologies according to recent white papers of Motorola [109] and Android [6][8,9,10]. The rates are expressed in Kilobits per second (kbit/s) (remember that 1 Mbit/s equals $2^{10} = 1024$ kbit/s). Similar values have been experienced by own validating experiments, performed in the context of the work for [104]. It is reminded that fast wired connections, e.g., Gigabit Ethernet, are still many orders of magnitude faster. Cellular networks are often assigned to the category of wireless Wide Area Networks (WAN). However, WANs do not only include cellular networks, which are of interest for this work, but also broad-area ad-hoc networks, such as Wide Mesh Networks (WMN), which are based on the 802.16 standards and are based on ad-hoc connections and distributed routing algorithms that can be jointly optimized for big sets of nodes [88]. Although cellular networks are associated with devices used in telephony, it must be mentioned that they are also supported and used by a steadily increasing number of mobile devices (e.g., laptops, tablets, or other, smaller, specialized devices) that do not belong to (or not originate from) the domain of telephony.

WIRELESS PERSONAL AREA NETWORKS also play a role in mobile computing and mobile Web service usage, as they may also be used by devices in order to grant access to a network over other, closely located, devices. Examples of this category are infrared and bluetooth, which are usually used for simple device-to-device connections. Simple device-to-device connections that are used for trivial actions, e.g., file transfer, and not in order to grant access to other networks are, however, not interesting in the context of mobile Web service usage.

Transfer data rates and delays are not the only interesting characteristics of wireless access networks, though. In any case, these are only the delays introduced by the cellular network technologies themselves, while the delays of end-to-end connections of devices with particular hosts depend on other factors as well, e.g., geographical locations [61]. Other characteristics should be considered, as well, when developing or

8 All data transfer rates mentioned in this section are downstream rates, i.e., for transmitting data from the network to the device. Upstream rates (the opposite) are always lower, but usually less important.

9 The values for HSPA refer to the version HSPA+ Release 8 (cf. [109]).

10 The values for LTE refer to a spectral bandwidth of 10MHz and in the presence of multiple antennas (2) at both the transmitter and receiver side (cf. [109]).

adapting systems for usage in particular wireless networks. For example, as explained in [11], the energy consumption models of devices change when communicating over different wireless access networks. Thus, when the reduction of energy consumption is a goal, the related characteristics of the networks should be taken into account. Pricing models are another example, as they are also usually different for different wireless access networks.

## 2.3 OTHER NECESSARY FUNDAMENTALS

Although this thesis is primarily positioned in the SOC and mobile computing domains, the subsequent chapters can be better understood when supported by a basic knowledge of some other fundamentals, which will be very closely related to specific goals and exact approaches and techniques that will be used. QoS and QoE, caching, and decision support with missing data are such fundamentals. The following subsections provide the necessary background.

### 2.3.1 *Quality of Service and Quality of Experience*

QoS and QoE are high-level terms that need to be specified before they can be used as metrics in the context of a particular domain or problem. As a result, many different definitions and many different subordinate aspects and parameters have been proposed for them.

For example, [133] defines QoS as "the well-defined and controllable behavior of a system with respect to quantifiable parameters". More closely related to the context of this thesis, [122] and [31] include models that explain which aspects and parameters QoS embraces when examined in the SOC domain, while [83] examines the QoS issues of Web services. Analyses of QoS attributes and metrics from the multimedia domain are also relevant, especially those that include and examine aspects of mobile computing. Such analyses are provided in [149] and [92].

While QoS can be measured, tested or evaluated without involving the user of the system (or application), QoE has emerged more recently as a quality-related concept that involves the user. One definition of QoE came from the International Telecommunication Union (ITU), claiming QoE to be "the overall acceptability of an application or service, as perceived subjectively by the end-user"[96], while Telenor gives a similar definition in [50], although it states that it is "based on both subjective and objective measures". QoS and QoE have been examined in parallel in [148], namely in the context of cellular networks.

From the point of view of this thesis and based on the above sources, the scope, the subordinate aspects, and example metrics of QoS and QoE can be summarized as follows:

QUALITY OF SERVICE refers to a set of technical aspects such as performance, flexibility, scalability, reliability and more, which can be used for characterizing the overall quality of a system, service, or application. For each of these aspects, case-specific metrics can be defined. For example, the data transfer rate is a performance-related QoS metric for wireless access networks (cf. Section 2.2), while response time is a common performance-related QoS metric for Web services.

QUALITY OF EXPERIENCE refers to a set of user-related aspects such as opinion, satisfaction, QoS-perception and more, which can also be used for characterizing the overall quality of a system, service, or application. Obviously, QoE depends on QoS, but it rather measures the *effect* that the QoS metrics have on the user. User ratings or user choices are examples of possible QoE metrics.

In the context of this thesis, QoS is important because it is the target of the adaptation mechanisms. Certain QoS metrics will be defined as goals (and later as evaluation metrics) of the developed adaptation mechanisms. Both QoS and QoE will play a role in the analysis of decision support by indicating two different approaches for taking adaptation decisions, namely QoS-based adaptation and QoE-based adaptation.

### 2.3.2 *Caching*

*Caching* is a technique that aims to improve the QoS or reduce the costs of certain computing tasks by storing data as close as possible to a data consumer that may need it again. The memory that lies close to the consumer, where the data is –usually temporarily– stored, is called *cache*. In the following, first the characteristics of caching approaches and, then, the metrics that are used to analyze and evaluate them, are discussed.

#### *Characteristics of Caching*

Caching may be performed in many different domains, such as the World Wide Web (WWW), i.e., for Web content [12], multiple-hierarchy disk storage [146], databases [5], Web service responses [159], and more. Furthermore, there are many possible goals that may be pursued with caching, e.g., reduction of response times, relieving of the network, reduction of costs. Different goals may be served better by different caching approaches.

Another important aspect that characterizes caching approaches is the principle on which their cache logic is based. The cache logic usually refers to the algorithms that are used to determine which data shall be cached, when cached data should be used, and when it should expire or be replaced. Although such algorithms in related literature often fall under the category "cache replacement strategies" [117], the term "cache logic" is more general and does not necessarily refer to "replacement", but rather to the core idea behind a caching approach, e.g., to the features (age, size, external information, etc.) on which the usage (or not) of cached data depends.

Caching approaches can also be categorized according to the system parts that are involved or have to be implemented. In *client-based* (or *client-side*) solutions, all the work is performed by the client system itself, while other system parts do not even need to be aware that caching is applied. *Server-based* (or *server-supported*) solutions rely among others upon logic that is implemented at the side of the server, while *proxy-based* (or *mediator-based*) solutions use intermediary systems in order to cache data or support the cache logic. As a matter of course, client systems are very often involved, i.e., contain some caching logic, even in proxy-based or server-based solutions.

Examining the possibilities for the *application domain*, the *goals*, the *involved system parts*, and the *cache logic*, the landscape of the characteristics of caching approaches is described in Figure 3, based mainly on the information and the opinions found in

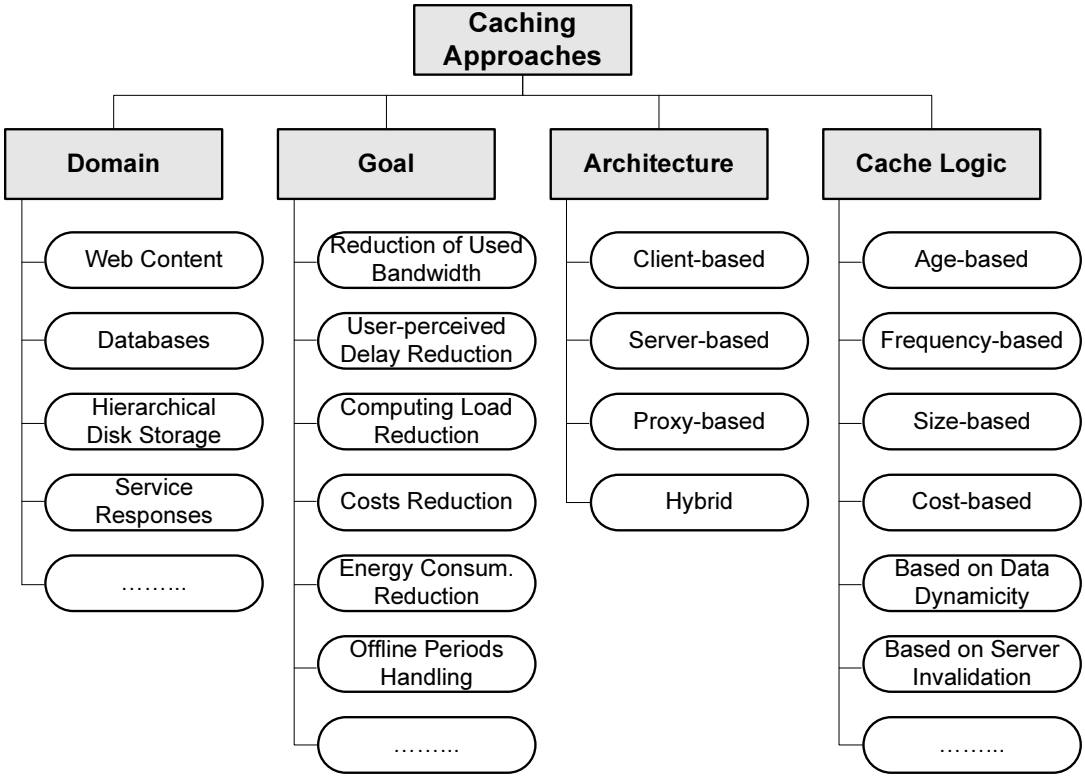



Figure 3: Categorization of caching approaches according to four different criteria

[12, 44, 117, 159]. It is, of course, possible, that many instances from each category are true for certain caching approaches.

*Metrics of Caching*

Independently of their application domain and their further characteristics, caching approaches are almost always characterized, among others, by two caching-specific metrics:

- The *cache hit ratio* (r) refers to the quota of the requests of the data consumer that are satisfied by the cache.
- The *freshness* (f) refers to the quota of the cache-served requests for which the used cache data are consistent with the data of the original data source, i.e., the quota of the cache responses that are up-to-date. Thus, freshness is a metric of consistency.

These two metrics are used in order to measure the *performance of the cache*. The latter should not be confused with the *performance of the system*, which is an end goal of the application of caching techniques. Enhancing the performance of the cache does not necessarily mean enhancing the performance of the system, although this is usually the case. Another term common to many caching systems is Time-To-Live (TTL), which indicates for how long cached data should be considered to be valid, and thus kept in the cache and reused. The term may be, however, misleading, because TTL is not necessarily a time constant, but it can be rather determined by complex

multi-variable functions. In the following paragraphs, let $p$ denote the performance of the system and $ttl$ denote the TTL value.

When caching systems are properly designed, the following normally holds for the relationships between $ttl$ and $f$, and between $ttl$ and $p$: $ttl$ is *inversely related* to $f$ and *directly related* to $p$. The exact relationship is case-specific. For example, $ttl$ and $f$ can be:

- linearly inversely related, i.e., $f = a - b \times ttl$ ($a$,$b \in \mathbb{R}$ constants).
- exponentially inversely related, i.e., $f = a - ttl^b$ ($a$,$b \in \mathbb{R}$ constants).
- or otherwise inversely related, e.g., sublinearly inversely related.

Analogously, $p = a + b \times ttl$, $p = a + ttl^b$, or similar. In any case, the above relationships manifest that there is a *trade-off* between performance and consistency (cf. also [69, 153]), when caching is applied.

### 2.3.3 *Decision Support and Missing Data*

Although decision problems are usually defined as the problems that can be answered with *yes* or *no*[11] and decision algorithms are the algorithms used to solve them, the term *decision support* is less deterministically defined and is more general.

Decision Support Systems (DSS) are not limited to including decision algorithms, but they rather usually include algorithms that are helpful for rating, ranking, scoring, or evaluating different alternatives. The base methods and the tools for developing such algorithms can be found in the fields of statistics, machine learning, and operations research [51, 170]. Optimization problems [51], Bayesian statistics [170], and decision trees [170] are examples of such basic methods.

Diverse issues, such as complexity and programmability, make the mentioned methods preferable (or not) under different circumstances. Another issue related to decision support approaches that act upon datasets of non-trivial size is their behavior in the presence of missing data, as well as the way the latter are handled. The following paragraphs provide a brief introduction to data missingness, because Chapter 6 will examine how the decision support algorithms presented in this thesis can be improved with regard to the handling of data missingness.

#### *Data Missingness and Imputation Algorithms*

Real-world data sets are often not complete, but they rather lack certain values that could not be obtained during the data collection process due to transmission errors, reluctance or inability of the data source to release the data, etc. Many researchers and other data analysts have been confronted with missing data, so that a theoretical basis has been developed in the field concerning the characterization, the classification, and the handling of missing data. In the following, the fundamentals of this theoretical basis are summarized, mainly based on [4, 90, 126, 131].

Assume a dataset $u$ of data units $u(n)$ and size $N$ ($n \in [1,N], n \in \mathbb{N}$). Each data unit consists of $J$ attributes $j$, $j \in \mathbb{N}$. Thus, each tuple $(n,j)$ describes one variable,

11 http://www.ccs.neu.edu/home/rjw/csu390-sp06/LectureMaterials/Decision-Problems.pdf (Last accessed in January 2012).

with the value $u(n,j) = v$ , $n \in [1,N]$, $j \in [1,J]$, from a predefined set of values $V(j)$ for each attribute. For modeling missing data inside such a dataset, the fact that a value could not be obtained needs to be stored. Hence, for each variable $u(n,j)$, an indicator variable $r(n,j)$ can be added, which is set to 1 if the value is present, 0 otherwise. $r(j)$ denotes the set of $r(n,j)$ for all $n \in 1..N$ and $r$ denotes the set of $r(j)$ for all $j \in 1..J$. The missing values of variables of a dataset is also referred to as *missingness*. The case where all variable values of a data unit cannot be obtained is called *unit nonresponse*, whereas the case when only a few variable values are missing is called *item nonresponse*. If the dataset is the outcome of repeated data collection from the same set of sources, a certain source may stop the data output for some time or forever. This phenomenon is called *wave nonresponse*, in case of "forever" it is called *attrition* or *dropout*.

The properties of missingness can be specified further by looking at what is called the distribution of missingness, that is, the distribution of the $r(j)$. Let:

- $u_{obs}(j)$ be the variables of attribute $j$ where the values could be obtained (or observed), that is, $r(n,j) = 1$ for all elements of $u_{obs}$.
- $u_{mis}(j)$ be the missing values, that is, $r(n,j) = 0$ for all elements of $u_{mis}$. Hence $u = u_{obs} \cup u_{mis}$, $u_{obs} \cap u_{mis} = \emptyset$.

The classification relates to the distribution of missingness as follows:

- The class Missing Completely at Random (MCAR) includes the cases where the distribution of missingness is independent of $u_{obs}$ and $u_{mis}$. That is, the distribution of the observed and the missing values have no influence on the missingness. In probability theory, two random variables A and B are independent if the following statement holds: $P(A|B) = P(A)$. This means that the probability that $A$ is true when it is given that $B$ is true, is equal to the unconditional probability of $A$ to be true. Hence, $P(r)$ is MCAR if $P(r|u) = P(r)$ holds, hence $P(r) = P(r|u_{obs})$ and $P(r) = P(r|u_{mis})$. Both, $u_{obs}$ and $u_{mis}$ not correlated to the distribution of the values of the complete data set.
- Missing At Random (MAR) is a class of cases where the distribution of missingness depends on observed values, but not on missing values. Hence, $P(r)$ is MAR if $P(r|u) = P(r|u_{obs} \cup u_{mis}) = P(r|u_{obs})$. For a single random variable, that means that its missingness distribution may depend on values of some other random variable, but not on its own. When looking at the response behavior, MAR means that for one data unit $u(n)$ that is requested from a data source, the ability (or decision) of the data source to not reveal the value $u(n,j)$ of a variable may be influenced by other observed values from the set $u_{obs}$, but neither by the required value $u(n,j)$ itself nor by variable values from the set $u_{mis}$.
- The last class of missingness is Missing Not at Random (MNAR), which denotes distributions of missingness that are neither MAR nor MCAR. That means that the missingness *might* be associated to any other values, missing or observed. Observed At Random (OAR) is a subclass of MNAR, where it is known that the missing data have no association with observed data.

Note that MCAR has the strictest requirements in terms of correlation between the data and the missingness, followed by MAR and finally MNAR, which does not have any restrictions at all. As the three classes are mutually exclusive, each missingness found can be described by exactly one of the classes. The described classification works for complete data sets as well as for single random variables, thus allowing to treat them individually if needed. When working with real data, the classification of the distribution of missingness is difficult. Since in most cases the real distributions of the data are unknown, it is impossible to test whether a given data set is MAR or not [131].

Once the theoretical basis has been provided, different approaches have appeared for handling missing data. Most of them fall under the category of "imputation algorithms", because they substitute missing values with other values, which are expected to be close to the original ones or to affect the further data processing as least as possible. Some of the most commonly used approaches are summarized here, while one of the most complete lists of such approaches can be found in [87], along with more detailed descriptions of the underlying mathematical models.

- *Case deletion* is an approach according to which data units that miss values are removed from the data set, i.e., for all $n \in [1,N]$: if $\sum_{j \in J} r(n,j) < J$, then $u(n)$ is removed from the data set.

- *Modus Imputation* is an approach according to which the value that appears most frequently among the observed values of an attribute (*modus*) is used in order to replace all the missing values of that attribute. *Mean Imputation* is a similar approach, which uses the arithmetic mean instead of the modus.

- The *Random Hot Deck* approach replaces missing values with an observed value that is randomly chosen from the current dataset.

- *Distance Function Matching* is an approach based on the calculation of distances between data units. The logic is similar to that of Random Hot Deck, but the value is chosen with higher probability from data units that lie spatially or temporally close to the data unit of the missing value. In the deterministic variation of the approach, the value "closest" to the missing one is always used for the replacement.

- *Multiple Imputation* is considered to be state of the art among the approaches that are based on the *maximum likelihood* concept. With Multiple Imputation, standard statistical methods are applied repeatedly in order to calculate estimates and confidence intervals of the examined variables, so that a certain degree of uncertainty exists during the replacement of missing values and certain statistical characteristics, such deviation, are not lost from the final dataset. For more details, refer to [4, 127].

As already mentioned, it can be rarely judged without further experiments which approach suites best a given problem, because the success of imputation depends on the characteristics of the missingness, on the values of the dataset, but also on the nature of the problem and on the algorithms that will run upon the dataset after the imputation.

# 3

# ADAPTATION IN SERVICE-ORIENTED COMPUTING

»The proof of evolution lies in those adaptations that arise from improbable foundations.«

— Stephen Jay Gould

System adaptation is the general approach that will frame all the enhancements proposed throughout this thesis. For this reason, this chapter is dedicated to offering a transition from theoretical adaptation concepts to practical ideas about what kind of adaptation is relevant for SOC and what potentials arise if this adaptation is performed with a focus on wireless participants. After a general introduction in adaptive systems, an important notion of adaptive service-oriented systems, namely autonomic computing, is presented. The idea of enhancing service-oriented systems through adaptation is then validated with an own extension of a state-of-the-art open source service platform, while the chapter concludes with a discussion of the potentials of drawing the focus of SOC adaptation on wireless participants.

## 3.1 self-adaptive systems

The scientific field concerned with adaptation and *self-adaptive systems* is not based on completely new ideas, but it rather borrows concepts from other fields, such as biologically-inspired computing, distributed artificial intelligence, machine-learning, control theory etc. [25]. However, the ad-hoc combination of isolated approaches from these different fields is not enough. Therefore, more specialized software engineering approaches and runtime maintenance techniques for developing self-adaptive systems are emerging [25, 129]. A self-adaptive system is a system that adjusts itself or its behavior to its environment. Many questions about what, where, how, and when should be adapted arise. Although the answers depend heavily on the system, there are some common denominators that help develop generic frameworks and approaches, which can be reused for different self-adaptive systems.

One of these common denominators is the so called *generic adaptation loop*. The generic adaptation loop is not only one of the most important ingredients of adaptation, but also one of the best ways to explain the adaptation concept. It is the formal representation (or the engineering expression) of the fact that a system adjusts itself to its environment and it should be revisited or refined for every self-adaptive system. Such a refined version will also be presented in the context of this thesis, in Chapter 4. In its generic form, the loop has appeared in different versions, which are nevertheless very similar.

Figure 4 shows two different versions of the generic adaptation loop, based on [129] and [28]. In both versions, the lifecycle of a self-adaptive system is split into four phases. The theoretical viewpoints behind these versions may differ in some details. For example, [129] considers the recognition of system symptoms as part of the first phase (monitoring), while the first phase of [28] seems to be limited to the pure

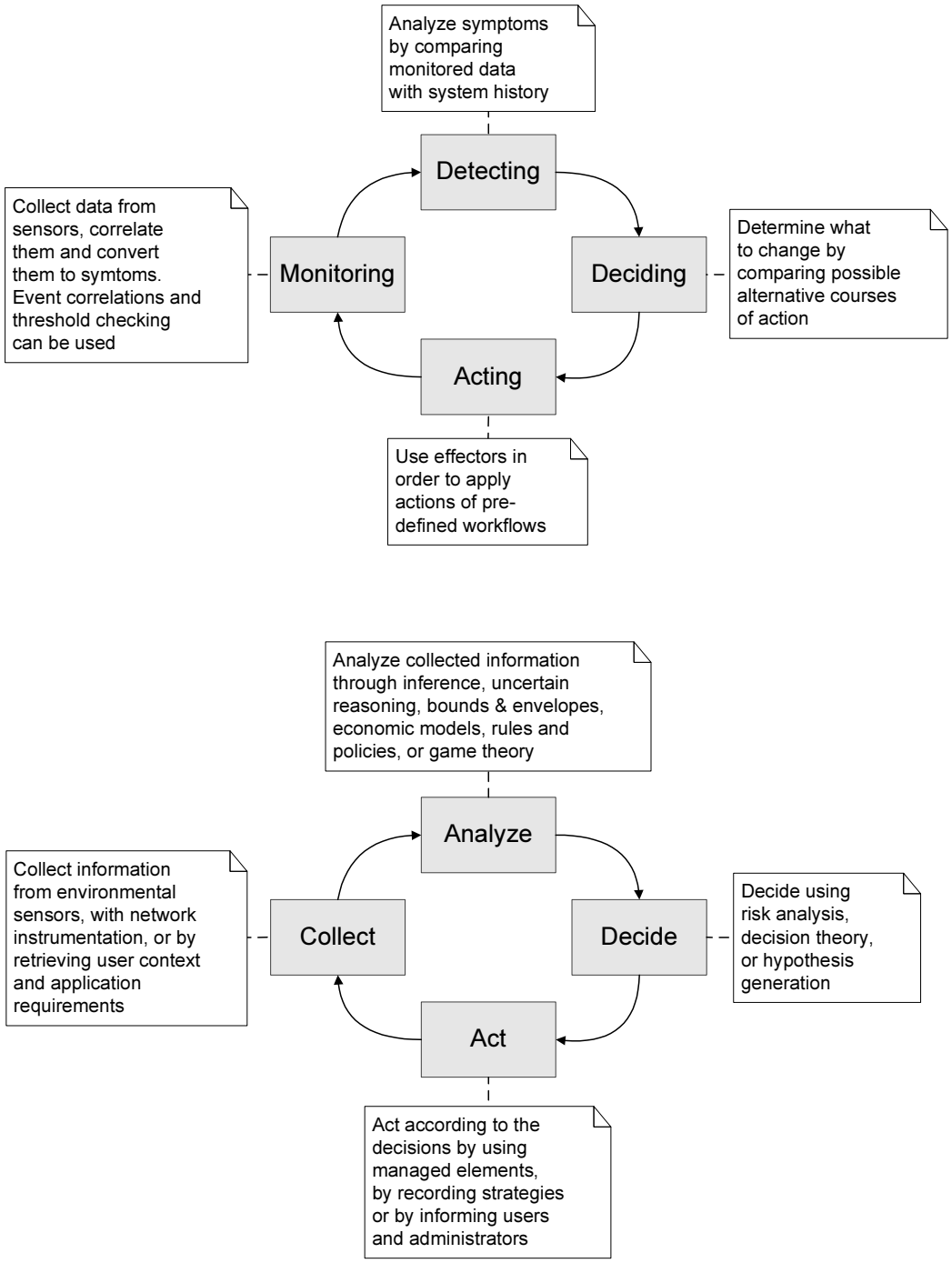



Figure 4: Different versions of the generic adaptation loop, based on [129] (top) and [28] (bottom)

collection of the data. Further differences become obvious by studying Figure 4 or the referenced publications. However, a one-to-one matching of the adaptation phases, as well as of their core concepts and employed techniques, can be easily performed.

As the above lifecycle can have different instantiations and may be used for different purposes, a set of properties (often referred to as *self-* properties* has been defined for self-adaptive systems, in order to describe the nature (or the purposes) of given instantiations. The most important of these properties [62, 129] are listed here because they will be referenced in subsequent sections:

SELF-HEALING is the property of identifying, diagnosing, and repairing problems, bugs, or other failures, which appear during the operation of a system, probably due to changes of its environment.

SELF-OPTIMIZATION is the property of modifying particular system parts in order to enhance system performance. The fact that runtime modifications can enhance performance is, again, because the context of the system usage can change dynamically. Optimization here does not have the meaning of "finding an optimal solution" (as in the field of operations research), but it could be a synonym of self-enhancement.

SELF-CONFIGURATION is a property that relates mostly to incompatibilities between system parts or software modules. It is understood as the re-installment, updating, or re-arrangement of them, so that the system remains functional despite dynamic extensions or revisions.

SELF-PROTECTION is a security-related property, which means that the system can prevent the propagation of errors than cannot be self-healed or even that it can enhance its security mechanisms by "learning" throughout its operation and without externally added enhancements.

SELF-ORGANIZATION AND SELF-MANAGEMENT , when used in this context, are normally superordinate terms, which refer to the whole set of the aforementioned self-* properties. In a more general context, self-organization is often understood as a synonym of "biologically inspired computing". [85] offers an interesting comparison of self-organization mechanisms in nature and IT.

## 3.2 SELF-ADAPTIVE SERVICE-ORIENTED SYSTEMS

A variety of research works related to adaptation have appeared in the SOC domain, as well. Some of them are more generally applicable, while others are concerned with very specific issues. They usually focus on different aspects, e.g., on new architectures for QoS-Management in SOC [164] (self-optimization), on new Web service specifications and standards for defining adaptation (re-)actions [123], on mechanisms for letting service-oriented systems automatically adapt to changing service implementations [161] (self-configuration and self-healing), or simply on re-visiting generic adaptation concepts in the context of enterprise SOC [14] (all self-* properties). Among the most important series of works in the field are the works on *Autonomic Computing*, which originate from the International Business Machines (IBM) company. In the following, the works on autonomic computing are summarized in order to give an example of how self-adaptation can be interpreted in the SOC domain. After that, a proof-of-concept for self-adaptation in SOC is provided. For the latter purpose, an open source service platform has been extended in order to include self-optimization features.

### 3.2.1 *Autonomic Computing*

The works of IBM on autonomic computing [14, 62, 108] focus on enterprise architectures, often have explicit references to particular SOC technologies, e.g., Web services [62, 108] or even the Web Services Distributed Management (WSDM) standard [14], and they consider all self-* properties. Actually, although they are no generic or pure theoretical works, they belong to the works that have established the term *self-**. More concretely, the works on autonomic computing propose and describe, among others, the following:

- The existence of a set of components inside the SOA (called *autonomic elements*), which have well-defined roles in a conjoint attempt to fulfill the adaptation lifecycle. *Autonomic managers* are hierarchically placed inside the architecture in order to control and orchestrate the lifecycle. *Managed resources*, *self-managing resources*, *knowledge sources*, but also *manual managers* help them achieve this.
- The combined use of a set of specialized SOC-related standards, namely WSDM, WSDM Event Format (WEF), a symptoms reference specification, and a solution deployment descriptor. The concepts are similar to popular WS standards, though these ones should not be designed for developers, but rather for automated machine-to-machine communication inside a self-adaptive system.

Although Autonomic Computing provides a SOC-specific view of the abstract adaptation concept, it is still an engineering concept that is far from presenting exact technological solutions and proving that self-adaptation in a service-oriented system can enhance performance. The latter potential will be examined in the next subsection through a more concrete example.

### 3.2.2 *Proof-of-Concept with an Open-Source Service Platform*

The goal of this subsection is to provide a proof-of-concept, namely to discuss some experiments that have been performed with an open source service platform. The service platform has been extended with self-optimization features that can help keep service availability at high levels despite changing usage conditions. The used service platform, the added functionalities, and the main evaluation results are briefly summarized in the following, while all design and implementation details have been omitted. For the latter, please refer to [101, 102].

*The service platform*

In accordance with the nature of service-oriented software, some tasks exist, which must be fulfilled in almost all SOA solutions, e.g.:

- The service registry mechanisms (publication of service descriptions, look-up of service descriptions etc.).
- The address resolution (mapping of name-based service invocations to exact addresses/endpoints).

- The service deployment (loading, configuration, starting, and stopping of services).
- The management and monitoring, usually in the form of auditing and logging, with focus on QoS parameters such as service response times or hardware metrics such as CPU load.

As already mentioned in Chapter 2, these tasks can be undertaken by a service platform (or an ESB). Thus, enriching a service platform with self-adaptation mechanisms means that these mechanisms will be present in every system built to run upon this service platform. The service platform that has been selected for experimenting with self-optimization extensions is Apache Tuscany[1], which is the state-of-the-art Service Component Architecture (SCA) platform. "SCA is a set of specifications which describe a model for building applications and systems using a SOA"[2]. No service platform has fulfilled the original vision of autonomic computing [29]. Thus, like all other service platforms, Apache Tuscany lacks many relevant features. The following paragraphs describe the main functionalities with which Apache Tuscany has been extended. Prototypical self-optimization techniques that are enabled by the added functionalities are then evaluated.

*Main added functionalities*

The extensions have been based, firstly, on the distribution of the platform core and, secondly, on the enhancement of the monitoring mechanisms. More concretely, the ability to perform deployment actions has been distributed to many different nodes, so that services can be, for example, started or stopped remotely. As for the monitoring modules, they have been enriched with Event Stream Processing (ESP) capabilities [81], so that any node can launch so called *software sensors* with a developer-defined goal in order to capture events that concern particular services. As a consequence, the extended platform is equipped with the following two functionalities:

- *Interest Registration*: Any component can register itself as "interested" in a service, storing at the same time its related ESP queries, determining this way what kind of information the software sensors will be sending to it and when. Components that register their interest contain software modules called *actors*, which enforce reactions upon the occurrence of certain events.
- *Service Instance Control Mechanism*: The extended deployment mechanisms offer to any component the possibility of retrieving the number of running instances of a particular service, as well as the addresses of the nodes that could potentially host further instances. Builds on these capabilities, the Service Instance Control Mechanism and can be used by any component in order to define a minimum number of instances of a service that should be running. This is obviously a service replication-based approach. A related approach, which, however, focuses rather on mobile service hosting, is proposed in [30].

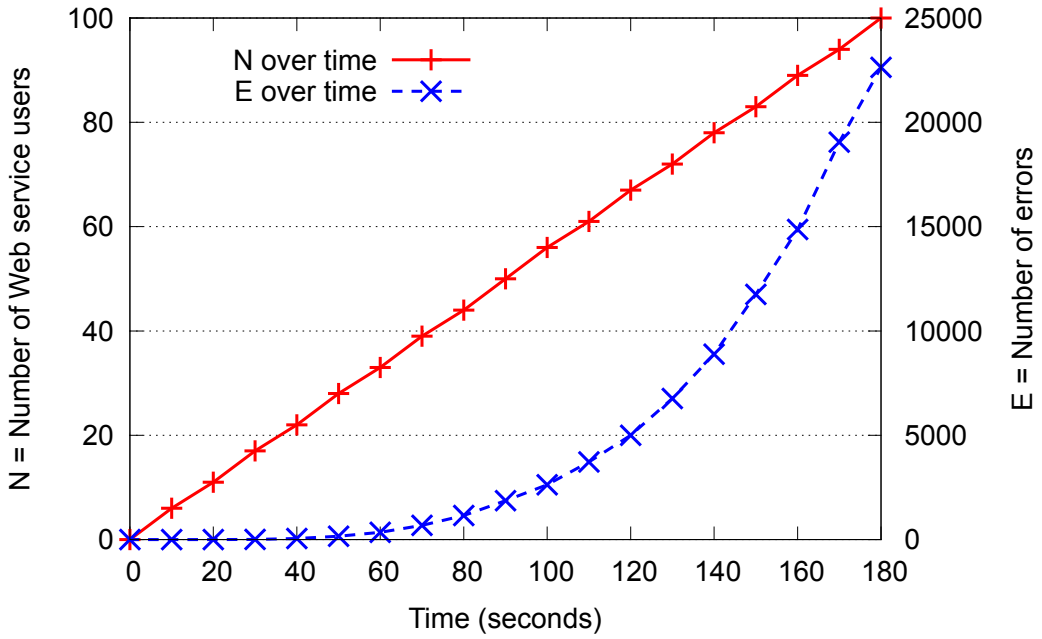



Figure 5: Input of the test runs (i.e., number of Web service users over time) and the corresponding erroneous service invocations (for a test run with a simple service)

*Evaluation*

The experiments have been performed on a simple, prototypically implemented, Web service that receives keywords and returns responses of 1-2 kilobytes, after searching in a database. This service, from now on denoted with S, has been subject to load tests[3]: External clients have invoked it with the pattern shown in Figure 5. A linearly increasing number of clients has been sending consecutive requests for 3 minutes. Figure 5 also shows how the linear increase of users leads to an exponential increase of erroneous service invocations, i.e., to an exponential decrease of the availability levels. The test-clients record errors when a timeout occurs.

The load of Figure 5 has been sent to S for four different scenarios. With $N_t(x)$ denoting the number of occurrences of $x$ in the last $t$ seconds, the *availability* of S is defined here as the value

$$A = \frac{N_{10}(\text{Successful invocations of S})}{N_{10}(\text{Invocations of S})} \times 100\%$$

This value has been measured over time for the following four scenarios:

- *Scenario1*: An instance of S is running on the original Apache Tuscany release platform.
- *Scenario2*: Three instances of S are running on the original Apache Tuscany platform and the invocations are equally distributed to them. The number

1 `http://tuscany.apache.org` (Last accessed in January 2012).
2 `http://oasis-opencsa.org/sca` (Last accessed in January 2012).
3 The load tests have been implemented with the tool soapUI, cf. `http://www.soapui.org` (Last accessed in January 2012).

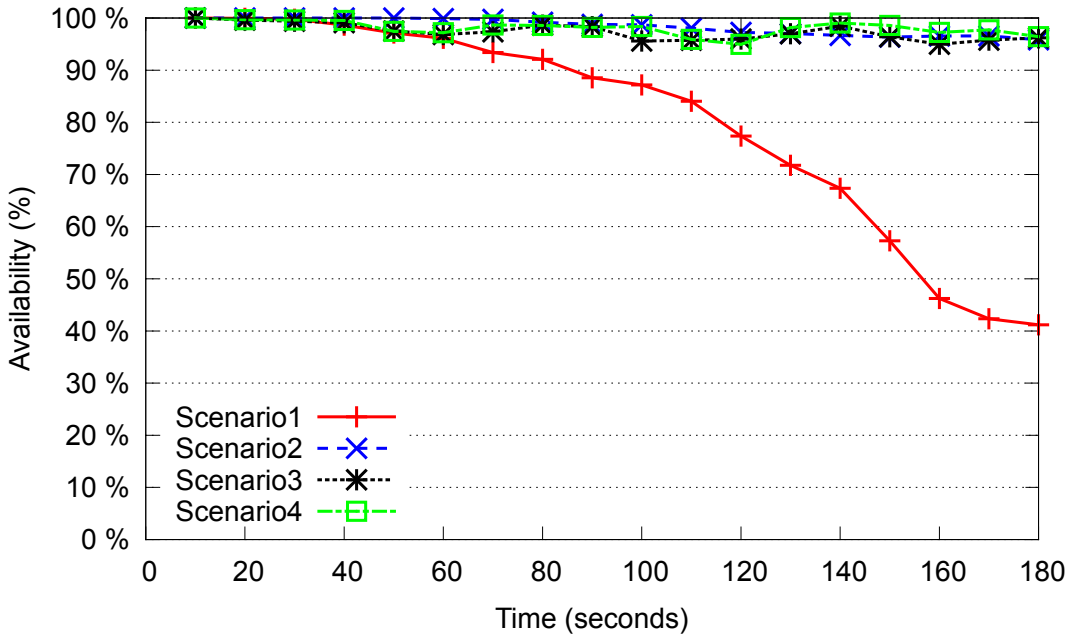



Figure 6: Availability of all tested scenarios over time

of instances (3) has been chosen empirically, so that it could almost always satisfy the given invocations' curve (Figure 5). For this case, as well as for the next two cases, the distribution of the invocations among the instances has been simulated. This is safe because the load balancing is not the focus of the experiments, though it would, of course, be interesting to test with different balancing of the invocations.

- *Scenario3*: An instance of S is running on the extended platform, the deployment instance of a node (more nodes could be used for fail-safety) registers itself as interested in S, with a query for retrieving the number of users of S each second. The deployment instance (more precisely its "actor" upon the retrieved data) has the following simple logic: use the Service Instance Control Mechanism to add an instance every time that the load of S exceeds a limit. This limit was chosen in this scenario so that, for the given input of Figure 5, the mechanism is started every minute.

- *Scenario4*: As in *Scenario3*, with the difference that the Service Instance Control Mechanism now doubles the number of instances every time it is triggered. With these two different configurations, the flexibility of the freely defined adaptation logic is shown, indicating how the platform can easily integrate application-dependent logic in order to be optimally exploited in different systems. Obviously, the choice of this logic affects the results.

Figure 6 and Figure 7 present the results for the four described scenarios (Figure 7 is actually a zoom into the three "good performing" curves of Figure 6). Although the experiment has been performed with a sample service, it is easy to understand that similar effects would be noticed for almost any Web service, maybe with slight modifications of the invocation pattern. Furthermore, this evaluation intends to demonstrate the possibility of enhancing a service platform through self-adaptation

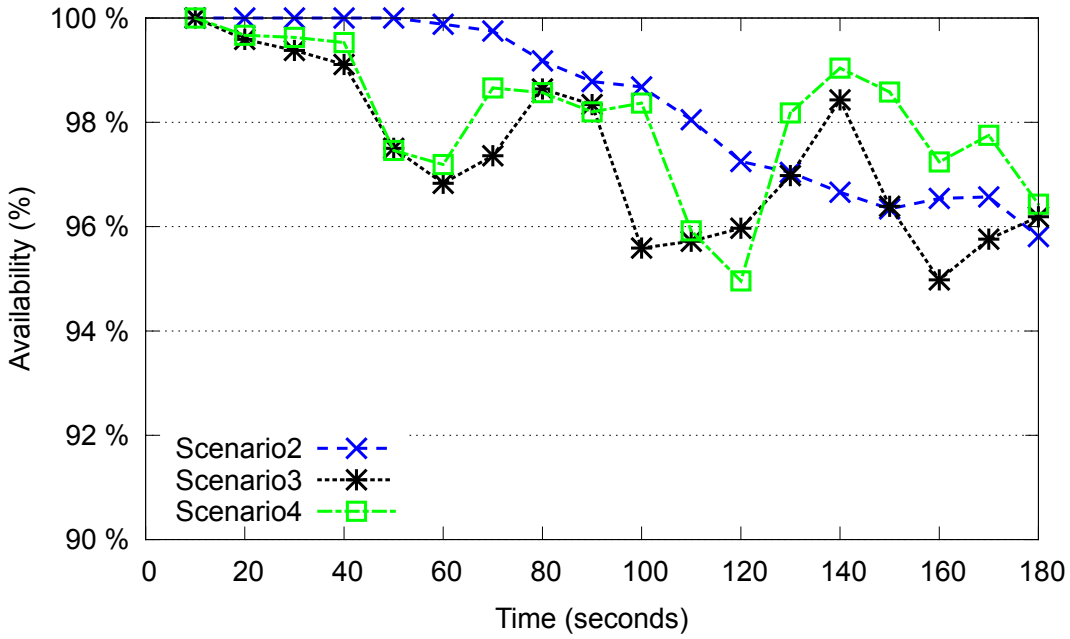



Figure 7: Availability of scenarios 2, 3, and 4 over time (Zoom in the high-availability results of Figure 6)

and should not be seen as a direct and complete comparison of different approaches (or platform versions).

The results for *Scenario1* prove that the availability of a service sinks when the number of users increases heavily. The same effect is slightly noticeable even in the case of the second scenario that is based on the original Tuscany platform, namely *Scenario2*, although the number of service instances was manually chosen in order to satisfy the given input. The decrease of the availability level is in that case much slower than in *Scenario1*, though steady. If the number of users would grow further, then the number of service instances would not be able to satisfy them any more, and an effect similar to that observed in the case of *Scenario1* would appear. Even if the maximum load that can be expected for a service is known from the beginning, excluding this way the possibility of such effects to appear, the usage of many instances from the beginning can lead to a big waste of resources. In scenarios where the service usage explosion is expected to happen suddenly but also rarely, this waste will be ongoing during most of the time.

Contrary to *Scenario1* and *Scenario2*, the number of service instances during *Scenario3* and *Scenario4* is adapted to the service load, maintaining high availability levels without wasting resources. Figure 7 shows the effect of service instance control. The component that uses the extended mechanisms in order to perform the service instance control is (implicitly) informed (in this case every ca. 1 minute) by its software sensors that the availability is sinking. Accordingly, further service instances are remotely deployed and the service invocations are again distributed among them. Thus, with an appropriate configuration at the side of the monitoring (and acting) component, the availability can be maintained at the desired levels, as long as this is allowed by the total resources that are available in the system. In a similar manner,

the service instances can be adapted to a decreasing number of users, though this is not shown in this evaluation.

During the last minute of the evaluation, *Scenario4* presents a higher availability, because the number of service instances is increased more abruptly. The difference between *Scenario3* and *Scenario4* demonstrates the configurability of the used mechanisms. The fact that different logics can be used inside the adaptation mechanisms offers flexibility in the regulation of the availability levels and of their trade-off with the costs. For example, a logic like the one used in *Scenario3* would be used in a scenario where service instance adaptations can be performed often, while the logic of *Scenario4* would rather be applied in scenarios where the frequent adaptation is either impossible or not desired.

As mentioned, the regulation of availability with self-adaptation mechanisms is only an example and a proof-of-concept, intending to demonstrate that state-of-the-art service platforms can be enriched with adaptation mechanisms and that this can, in turn, lead to significant enhancements. However, this is only a general statement and one could still think of a vast amount of different possibilities, but also constraints, concerning the application of the idea towards different goals. This thesis is concerned with enhancing Web service performance for wireless consumers. Therefore, the next subsection discusses the potentials of drawing the focus of the adaptation mechanisms on wireless service participants.

Further *related approaches* that can be considered as *technology-specific* instantiations of the self-adaptation principle have appeared in the SOC domain. Some of them consider fault-tolerance and, with that, service availability, as well [48, 65]. However, any comparisons of the presented service platform extension with them are out of scope, because the extension has been developed merely as a proof-of-concept and not as a solution for the mobile computing challenges that are of central interest for this thesis.

## 3.3 THE POTENTIALS OF FOCUSING ON WIRELESS PARTICIPANTS

As already mentioned in Section 3.1, the systems that can profit from adaptation are those that have variable, multifaceted, or changing environments. Compared to stationary participants of traditional IT systems, wireless participants are prone to exist in environments that present bigger variations in terms of device and network capabilities, user contexts, and more (cf. also Section 2.2). Therefore, a number of different potentials can be identified for the adaptation of SOC technologies for wireless devices:

- First of all, there is a big potential for adaptation mechanisms related to *performance* [80, 176]. The roots of this potential lie in the variety of network and device capabilities, together with the fact that, because of the nature of SOC, services are very often designed without prior knowledge about the types of consumers that will use them. For example, if a service is designed to communicate using a protocol/format that is appropriate for smartphones, then the ability to automatically adapt the used communication protocol/format would be worthwhile if the target group may change in the future. Performance is, in any case, usually the first "bullet" when listing the challenges of mo-

bile SOC. Specific solutions in this direction would fall under the category of *self-optimization*.

- A further potential stems from the fact that the technologies for implementing mobile service clients are not as mature as the technologies for implementing stationary or PC-based service clients [99, 158]. Therefore, apart from unexpected performance issues, *technical incompatibilities* may also appear when new types of mobile clients join a system. For example, if many mobile clients which are not equipped with implementations of certain specifications needed in order to process SOAP responses joined a system, it would probably make sense to adapt the service so that it can be addressed with simpler message formats. Specific solutions in this direction would fall under the category of *self-healing* or *self-configuration*.
- Last but not least, *context-awareness* is often a very important issue in mobile applications [8]. Even when the appearance or the definition of new types of context in a system does not cause performance degradation or incompatibilities for mobile clients, it may still be desired that the mobile clients become capable of using these new types of context and it is normally desired that this happens in an automated way. For example, [98] discusses, among others, how Web services can be automatically adapted in order to be able to handle optional parameters, which can be used for personalization and context-awareness. Such solutions would fall under the category of *self-configuration*, or even *self-optimization*, if Quality of Information (QoI) is considered to be an aspect related to QoS or performance.

*Examples*

Figure 8 illustrates[4] the three mentioned potentials with somewhat more concrete examples, though still high-level ones. Assume $WS_1$ as a location-based standard (SOAP) Web service that provides the exact locations of emergency vehicles, e.g., ambulances and fire tracks, along with plenty of information about their status and the operating staff. As input, $WS_1$ requires the location range of interest and an optional parameter which indicates the exact interests of the consumer and can thus be used in order to filter the results. Imagine that this service, which has been typically used by workstations stationed in the headquarters or offices of the emergency-handling organizations, is desired to be integrated in new mobile applications which support the on-site staff.

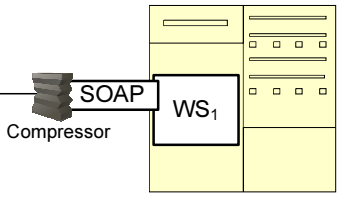
The Web service-related technological requirements that arise because of this new way of use have not necessarily been considered during the development or the previous operation of the service. Thus, adaptations to the needs of the mobile devices that will appear as service consumers are necessary. Ideally, these adaptations can be enforced in an *automated manner*, not only for $WS_1$, but for any similar service that needs it.

4 The example scenario is inspired by a project in which the author participated, cf. `http://www.soknos.de` (Last accessed in January 2012), while the screenshots are taken from own demonstrators of the author.


Web service WS1 offers an operation that informs about the locations of emergency forces (e.g., ambulances and firefighters). The operation takes a parameter that indicates the location of interest and an optional parameter that can be used to filter the searched types of forces. WS1 provides a SOAP interface and has been typically consumed by stationary devices.

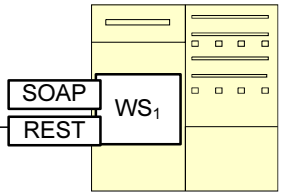
*If, at some point of its lifecycle, WS1 has to be used by...*

*a)...mobile consumers of type A, which has powerful processing capabilities but typically uses very low bandwidth connections:*

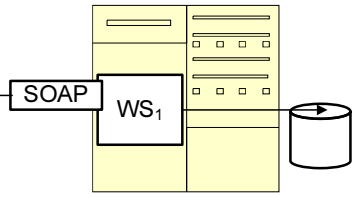

Compression of the responses can be (automatically) activated by the Web server of the service provider. **(Performance)**

*b)...mobile consumers of type B, which have no support for SOAP client development:*


The service can be (automatically) offered with an extra interface that uses, for instance, a REST message format, which can be consumed more easily by this device type. **(Technical Incompatibilities)**

*c)...mobile consumers of type C, for which information can be found in inter-application context databases:*


The service can use context data in order to (automatically) set the value of the optional parameter, which has not been sent by the consumer, thus filtering the results better. **(Context-awareness)**

Figure 8: Examples that illustrate the three main potentials (i.e., the handling of performance, technical incompatibilities, and context-awareness) of drawing the focus of Web service adaptation towards mobile consumers

The three cases shown in Figure 8a–Figure 8c provide a feeling about how a performance-related, a technical-incompatibility-related, and a context-awareness-related adaptation action could look like for this Web service.

Assuming that the big-sized responses of $WS_1$ cause low performance of the Web service calls when the latter are performed by a bandwidth-constrained consumer, Figure 8a shows that activating compression for the Web service responses could heal the performance problems. It should not be forgotten that, in the absence of mobile consumers such as those of Figure 8a, compression may not only be unnecessary, but also burdening for the (server- and client-) CPUs, and thus undesired [160].

Another possibility is that the new mobile application must be quickly developed or adjusted for a different type of mobile devices, whose platform does not offer native SOAP support, so that the implementation of service consumers for $WS_1$ involves a very high degree of complexity. For example, for Java-based mobile applications, Web service consumption has been enabled by the definition of a corresponding specification[5], which is not necessarily implemented on every device. In such a case, it would make sense to automatically offer the Web service with an extra interface and messaging format, e.g., REST, thus easing the Web service consumption on the devices of interest, as shown in Figure 8b.

Finally, it is possible that, in order to enhance the quality of the provided information, it is desired to filter the results of the Web service responses even for clients that do not provide the corresponding optional parameter. Although this may be impossible, if consumers such as those of Figure 8c appear in the system, for which related context information can be found in a separate context database, the automated enrichment of the Web service calls might be an option. Figure 8c demonstrates a case in which the Web service has been implicitly informed that the consumers of this type are only interested in information about ambulances.

Although this has only been an example, it must be noted that location-based services comprise a very important category of services that can be enhanced with adaptation mechanisms. This is because location-based services are present in a very big number of different application domains, they have a very close relationhip to context-awareness, and may exchange big amounts of data. Further, their consumption on mobile devices has only recently advanced, thanks to new positioning techniques, new programming languages specifications, middleware etc. [68]. All the above leads to a more frequent necessity for adaptation.

The three identified potentials are based on the study of related approaches and on considerations about the most important issues of mobile SOC (cf. also [99, 178]). However, this does not mean that the potentials of the application of adaptation mechanisms in the field will always be limited to these three points. New approaches may enable or reveal new dimensions of the enhancement of mobile SOC through adaptation mechanisms.

The current thesis will focus on performance, and, to be more specific, on adaptation mechanisms for enhancing the wireless performance of existing (possibly third-party) Web services. The next chapter will eventually focus on particular adaptation mechanisms of this category. However, before presenting specific solutions, it will present a complete architecture/framework for the adaptation of services for wireless participants. Although the framework and the architecture will include modules for

5 `http://jcp.org/en/jsr/detail?id=172` (Last accessed in January 2012).

automated context enrichment and –implicitly– also for the handling of technical incompatibilities, all further approaches and scientific contributions will be dedicated to the goal of enhancing performance.

## 3.4 SUMMARY AND CONCLUSIONS

In this chapter, generic concepts of system adaptation have been presented, followed by a gradual concretization of the issue towards SOC adaptation and, finally, service adaptation for wireless participants. On the way towards achieving the long-term goal of enhancing SOC for wireless participants, adaptation will be the general approach. More specific solutions will be presented in its context.

Self-adaptive systems have been introduced as the systems that adapt themselves to changes in their environment and it has been discussed why almost all self-adaptive systems are characterized by an adaptation loop, which is practically an instantiation of the generic adaptation loop (cf. Section 3.1).

A series of works on autonomic computing have been identified as the state of the art for applying abstract adaptation concepts in the SOC domain. In an attempt to test the applicability of adaptation concepts in a service platform –which is, from a technical perspective, the heart of service-oriented systems–, a state-of-the-art open source service platform has been accordingly extended. Core capabilities of the platform have been distributed, dynamic monitoring and event-based reactions have been enabled, and replication-based adaptation mechanisms have been evaluated. Referring to [102] for design and implementation details, this chapter has mainly presented evaluation results that demonstrate how service availability can be enhanced or regulated by using such adaptation mechanisms.

It has been highlighted that the described service platform extension is merely a "proof-of-concept", which only includes prototypically implemented mechanisms and does not consider wireless participants. However, the chapter has concluded by explaining that, in the field of service-oriented systems with wireless participants, adaptation mechanisms can be mainly used in order to handle issues of performance, incompatibilities, and context-awareness.

# 4

# MEDIATION-BASED WEB SERVICE ADAPTATION FOR WIRELESS DEVICES

»There is something different about communications that are mediated by a piece of technology; it is easier to talk about difficult subjects, and that is both good and bad. You don't see the person's upper lip tremble. You don't hear their voice quiver.«

— Amanda Lenhart

Different service adaptation mechanisms can be better examined and compared in the context of an adaptation framework. Therefore, this chapter starts by introducing the high-level scenario of the work at hand, namely the Internet of Services (IoS). Then, the Mobility Mediation Layer (MML) is presented as an architecture for hosting service adaptation mechanisms in the IoS. After identifying the part of the MML where the adaptation mechanisms considered in this thesis reside, technical details of the considered contexts and a revisited adaptation loop (cf. Chapter 3) complete the description of the adaptation framework. Although the MML is referenced throughout this thesis as the host of the performed adaptation, all the mechanisms and solutions contributed by the thesis could be part of any layer similar to the MML, provided that it shares with MML the goals and, coarsely, the adaptation loop described in Section 4.3.

Once the framework and the focus are well defined, possible service adaptation mechanisms can be examined. Thus, Section 4.4 contributes a detailed study and comparison of approaches for lightweight Web service communication in the context of mobile computing, because these are the approaches that could be possibly used as adaptation mechanisms in the MML (or a similar layer).

## 4.1 THE INTERNET OF SERVICES SCENARIO

The IoS [21, 22, 95, 136] should be understood as a future scenario for service-orientation. In short, the IoS refers to a globalization of service-oriented solutions, where Web services are offered by different providers through common global service marketplaces. New stakeholders and roles (such as marketplace hosts or other intermediaries) appear [136], making the relationships between service providers and service consumers eventually even more loose and flexible. Universal service markets have been envisioned and discussed even earlier, e.g., in [77], as the common denominator of the collection presented in that work. However, with SOC still being immature, [77] focuses on telecommunication services.

The realization of the IoS scenario is supported and accelerated by certain enabling technologies. For example, new service description specifications such as USDL [22], which include the business and operational aspects in addition to the technical details, turn Web services into perfectly tradable goods. However, in the context of this thesis, more important than the technologies are some features of the IoS scenario. The main

*IoS features that determine how, where, and what kind of service adaptation can be performed* inside it, are the following:

- *Many Web services gathered under single portals*: If service marketplaces offer homogeneous and easy access to a large number of Web services, then particular adaptation mechanisms can be performed at once for many of them. Generic adaptation mechanisms (i.e., adaptation mechanisms that can be applied to any service) whose implementation or operation is difficult are probably not worth their effort when they can only be applied to few services. If the modules that can apply these mechanisms have access to global service marketplaces or even exist as marketplace extensions, then their development and operation may be much more beneficial.

- *Less predictable Web service usage characteristics*: Traditionally, Web services have been developed for a set of consumers which has been more or less known a priori. In a scenario where Web services are published as tradable goods waiting for potential consumers to find and buy them, it is much more difficult to predict under what system conditions the Web services will be actually used. For service adaptation mechanisms of the IoS, this means that they should consider a wide spectrum of different possible system contexts/conditions. This unpredictability is a side-effect of flexible relationships between service providers and consumers.

- *Less control or influence over third-party Web services*: Another side-effect of the looser relationships between service providers and consumers is the fact that consumers have less influence on the implementations of third-party services. This has the following very important effect on adaptation: Adaptation mechanisms that need any kind of modifications in the code or the hosting system of the service, which have often been examined in the context of SOC adaptation, normally do not come into question in the case of the IoS, because it is rare to assume such access rights. As mentioned earlier, the modules that host the adaptation mechanisms are expected to lie inside the control sphere of an IoS stakeholder other than the Web service owner.

The next section describes an IoS-oriented architectural solution for performing service adaptation for wireless participants.

## 4.2 THE MOBILITY MEDIATION LAYER

The MML has been conceived as a service adaptation layer that suits the IoS scenario, i.e., it has been designed for working with service marketplaces that give access to various services, it assumes no access to the implementations or the hosting systems of the services, and it focuses on the types of adaptation that are dictated by the needs of mobile computing (cf. Section 3.3). Such a layer could be operated by a service marketplace host that desires to offer services with different access methods or communication protocols, by a mobile application developer that desires to adapt the way services are consumed by her/his application, or by any stakeholder that has the business model of offering enhanced access to existing third-party services. Figure 9 shows the high-level MML architecture and indicates the modules where the

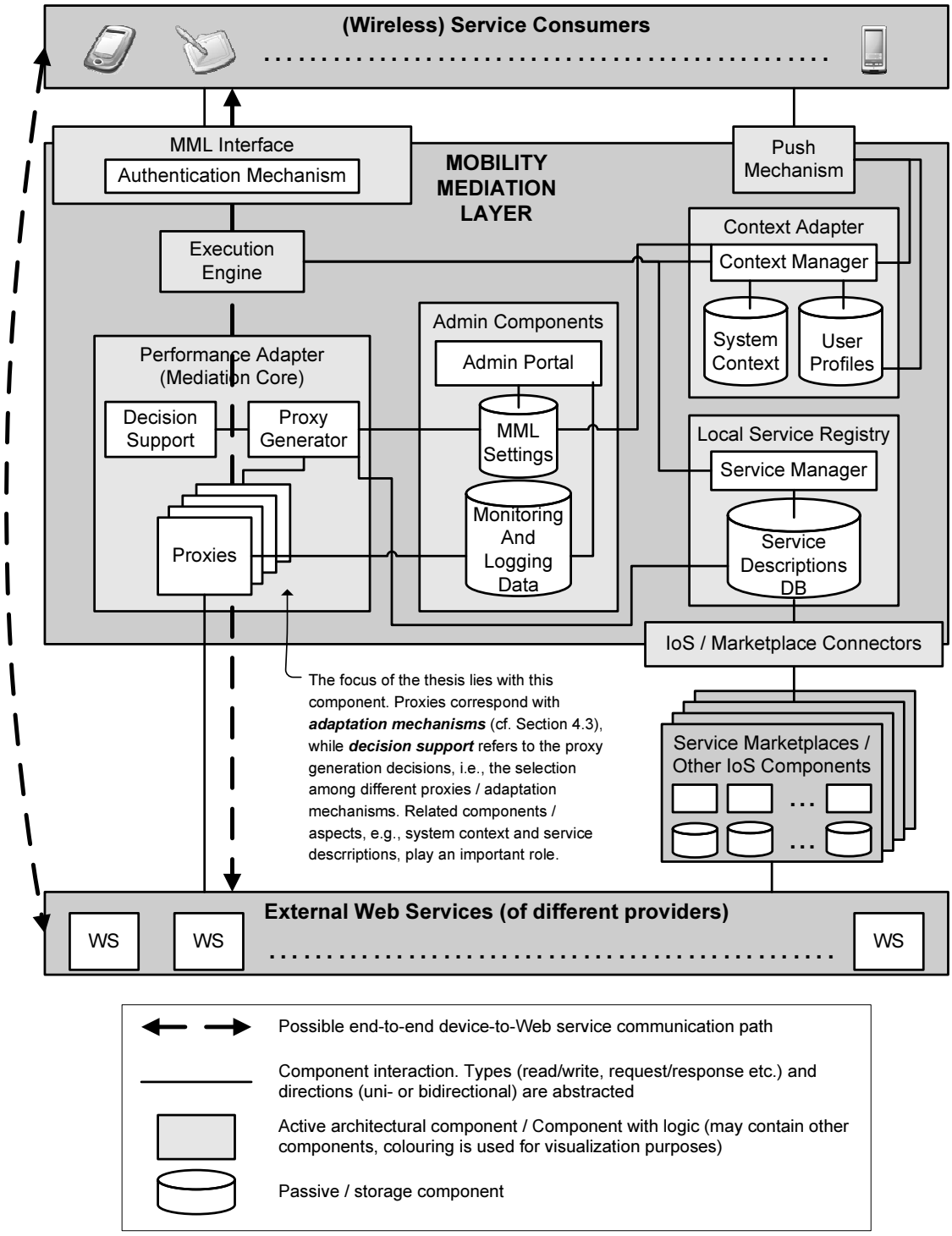



Figure 9: High-level architecture of the Mobility Mediation Layer

performance-related adaptation mechanisms are located. Before focusing on the latter, the complete MML architecture is briefly explained for the sake of completeness.

The MML can, of course, be accessed by all types of clients. However, its target group are mobile and wireless, often constrained, Web service clients. Thus, these clients can substitute their direct Web service calls (left dashed arrow in Figure 9)

with mediated calls through the MML (right dashed arrow in Figure 9). For this, they use the *MML Interface*, which also includes an *Authentication Mechanism*. The calls are then handled by the *Execution Engine*, which involves the *Context Manager* in order to personalize the calls and perform automated context enrichment and the *Service Manager* in order to find the appropriate services (and their *Proxies*) that will be used. Particular approaches for implementing or enhancing the application-spanning personalization and context-enrichment of the service calls are not examined any further in this thesis, because the focus lies on performance-related adaptation mechanisms. Interesting related considerations can be read in [15]. The actual call of the external Web service is then performed by a proxy, which may vary from a simple "dummy" request/response forwarder to the enforcer of any adaptation mechanism. More on this issue will be explained in the next sections. The availability of proxies, i.e., the types of proxies that are generated and used for each service, depends on the *MML Settings*, which, in turn, depend on the *System Context*, as can be seen in Figure 9, while the service descriptions are also necessary for the proxy generation process. Further, the MML includes an *Admin Portal*, which can be used to configure/administrate the MML or to manually examine monitoring and logging information stored by the proxies for every Web service call. Finally, the MML accesses service descriptions and other service-related information from the IoS through its *IoS/Marketplace Connectors*, while a *Push Mechanism* is responsible for the reverse communication with the wireless consumers, when the latter must be informed about an event that is triggered by the current context. Some of the MML components can be derived and/or are inspired by abstract components of general-purpose QoS-middleware concepts such as that of [91]. In particular, many conceptual similarities can be found between the MML adaptation components and the descriptions of [91] about QoS-adaptation with "resource adaptors", "component configurators", and "service configurators".

As already mentioned, the adaptation mechanisms presented in this thesis are not middleware-specific and could lie inside any other similar layer. For this reason, no further details about the MML as a whole are presented here. However, such details could provide a more complete understanding of the technical context. Therefore, Section A.1 provides further information and screenshots about the operation, the use cases, and the overall practical benefits of the MML. Furthermore, some initial ideas and the first high-level architecture of the MML can be found in [99].

## 4.3 FOCUSING ON ADAPTATION FOR PERFORMANCE

Before referring to exact adaptation mechanisms (or proxies), i.e., before examining what tasks and techniques can be performed by a proxy, the concept of using Web service proxies in order to enhance performance is first described abstractly. Thus, Subsection 4.3.1 explains the relationship between proxies and adaptation mechanisms and provides a simplified view of the mediated Web service communication, abstracting from all further details of a layer such as the MML. After that, all the background information needed for the design of a specific, revisited adaptation loop will have been presented. Thus, Subsection 4.3.2 provides and describes the revisited adaptation loop for the examined scenario.

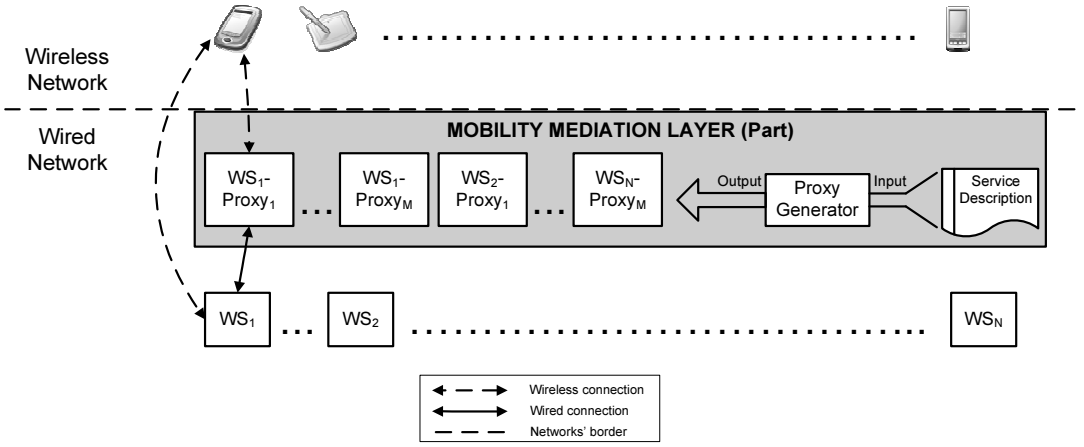



Figure 10: Simplified view of the Mobility Mediation Layer proxying concept

### 4.3.1 *Enhancing Performance with Web Service Proxies*

Due to the lack of control over external Web services (cf. Section 4.1) is the proxying of Web service calls the obvious approach for enforcing adaptation mechanisms [2]. Any layer that needs to mediate Web service calls would have to use some kind of proxies.

Proxying is an abstract concept that implies the interception of requests and it may be used in many different fields. The definition of a *Web service proxy* in the context of this thesis is specified with the following statement:

WEB SERVICE PROXY is a module of a mediation layer that can intercept the calls to a particular external Web service in order to enforce an independent and well-defined adaptation mechanism.

More concretely, as depicted in the simplified view of Figure 10, Web service proxies have the following mission: They try to avoid that a heavyweight communication takes place over the wireless channel by replacing direct calls (long dashed arrow in Figure 10) of wireless service consumers to Web services with proxied service calls. The latter consist of two parts: a wireless call (short dashed arrow in Figure 10) to the proxy and a wired call (short direct arrow in Figure 10) from the proxy to the Web service. As the proxy enforces an adaptation mechanism, the wireless call is expected to be more lightweight, i.e., to be completed by exchanging less data, while the wired call normally uses exactly the same data exchange as the original direct call (long dashed arrow in Figure 10) would. Different proxies enforce different adaptation mechanisms. The adaptation mechanisms that a Web service proxy can possibly enforce will be discussed in Section 4.4. Figure 10 also illustrates that multiple proxies may exist for the same service.

It must be noted that the Web service proxies could be implemented manually. However, such a manual implementation requires some development effort for every individual proxy generation. The approach of this thesis is an automatic Web service proxy generation that is based on the service descriptions (cf. Figure 10). This approach is much more challenging in terms of development, but, once implemented, it can be

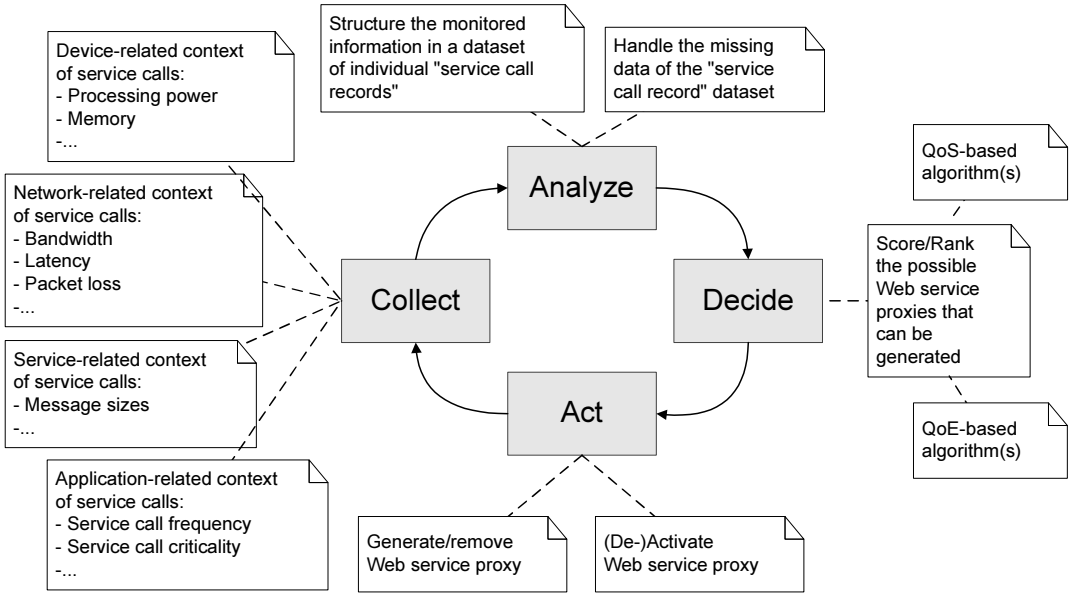



Figure 11: The case-specific adaptation loop for performance-related Web service adaptation in a mediation layer concerned with wireless service consumers

used for as many proxy generations as desired. The automatic proxy generation is also partly based on novel concepts and will be described in Chapter 5 as part of the solution for enabling a new adaptation mechanism.

### 4.3.2 *Revisiting the Adaptation Loop*

The previous sections concretized the examined case of adaptation, namely *performance-related Web service adaptation for wireless consumers*, so that a revisited, case-specific adaptation loop can be designed, as promised in Chapter 3. Referring back to the generic adaptation loop (Figure 4), the main case-specific contents of the four adaptation phases are listed in the following, while the resulting loop is shown in Figure 11. Low-level details, e.g., the exact and complete lists of relevant context attributes, can only be determined after the examination of the possible adaptation mechanisms in Section 4.4.

- *Collect*: Here belong all the attributes which can be potentially captured by a mediation layer and whose values may determine the adaptation decisions or actions. In the described case, these attributes could be either context attributes or Web service characteristics, while the latter may also be understood as part of the system context. Thus, there are four types of relevant system context: (i) Network-related context includes network connection features such as bandwidth, latency, packet loss etc., (ii) device-related context refers to the characteristics of the devices used by the Web service consumers, e.g., processing power, memory etc., (iii) service-related context refers to the features of the Web services, e.g., average request/response sizes etc., and (iv) application-related context includes application features such as the frequency of the Web service calls, their criticality etc.

- *Analyze*: In this phase, the collected context attributes must be preprocessed, structured, or analyzed in a way that renders them usable for the proxy generator decision support system. An important task of this phase is to gather the context data from the different sources and organize them in a structured data set of Web service call records. For example, one Web service call is characterized by Web service features, which can be found in the service registry or by testing the Web service, by monitored response times, which can be found in the monitoring and logging database, and lots more. Further, practical experience from using the MML in a project[1] proved that many of the information that can be potentially captured are missing for particular Web service calls. Therefore, a handling of the missing values through data imputation is an important part of the analysis phase.

- *Decide*: Once the collected data has been prepared, it serves as input of the decision support algorithms that help decide which proxies should be generated. By examining the possibilities for this case and referring back to the theory of Subsection 2.3.3, it becomes obvious that there are two possibilities for scoring the different alternatives: the "QoS-based scoring of the alternative proxies" and the "QoE-based scoring of the alternative proxies".

- *Execute*: The main actions involved in the examined adaptation scenario are the generation or the deletion of a Web service proxy, as well as its (de-)activation, i.e., the enforcement of its use (or not). Although the MML includes further actions, such as "register proxy", they are not considered to be directly related to the adaptation process.

The revisited adaptation loop also reveals the importance of context in the adaptation lifecycle. As it becomes obvious, context is handled at many different levels, as it is the object of the two first phases (*collect* and *analyze*), while it may also play a major role during the other phases. Thus, although some works ([82, 114]) have analyzed the research issues related to context in general, context may be examined from many different perspectives depending on the research focus. The perspective from which context is examined determines, in turn, the scientific methods that should be used for handling it.

For example, the adaptive system for the support of communication services presented in [134] could also be modeled with a similar adaptation lifecycle and presents interesting approaches for retrieving and evaluating context. However, the focus of that work is explicitly set on the retrieval of context and on its evaluation for identifying user stati. The user stati are then considered to be higher-level context information. The mentioned actions are parts of the first two phases. The current thesis focuses partly on the context analysis (with focus on data imputation), but more extensively on the decision support (Chapter 6) and the design and execution of adaptation actions (Chapter 5). Both the difference of the research focus and the difference of the system goals (capturing events vs. enhancing performance) render the specific algorithms used in works such as [134] out of scope for the current thesis. The same holds for works in the field of context-based multimedia content or rate adaptation [1, 124]. For example, [124] handles the network context as a "black

1 `http://www.greenmobility-project.de` (Last accessed in January 2012).

box" and focuses on involving user feedback as adaptation trigger, while [1] uses system context in order to design a set of predefined, video-specific, adaptation rules. Concerning the adaptation actions themselves (execution phase), the referenced works present technology-specific solutions, which are also out of scope for this thesis. All in all, ideas and approaches from works that involve context-based adaptation can be rarely and only partly applied, because the technological object of adaptation and the research focus vary in a very big scale.

## 4.4 STUDY AND COMPARISON OF WEB SERVICE ADAPTATION MECHANISMS

Given that a Web service proxy is a component that enforces a Web service adaptation mechanism and that it must be investigated which types of proxies come into question for the MML, it is necessary at this point to list and analyze the possible Web service adaptation mechanisms, i.e., the possible proxies. For this, the following subsections present the details and the results of a study and comparison of Web service adaptation mechanisms, based on related publications from research and industry. No similar survey has been found in literature that could form the basis for examining the described proxy-based adaptation. Furthermore, to the best of our knowledge, this is the first survey that examines and compares the adaptation mechanisms in the context of mobile computing, i.e., the first survey that discusses under which particular conditions of the mobile/wireless environment each adaptation mechanism achieves its maximum benefit.

The analysis of the benefits of different adaptation mechanisms under different circumstances can be useful in other scenarios, as well. Thus, the contribution of the presented survey is not limited to the scenario of Web service mediation, which has been described. For example, one can imagine future mobile applications that decide themselves at runtime which available service endpoint to connect to. Then, the insights of the survey would be useful at the side of the service consumer, as well. A service bus that offers multi-channel access [67] to its services is another example of a system that could exploit the insights of the survey.

Concerning the Web service adaptation mechanisms themselves, the related work will be covered in the next section. As for related studies, the work of [80] can be mentioned. Nevertheless, it only considers high-level Web service adaptation approaches rather than particular implementable adaptation mechanisms, it is limited to descriptions, and it does not offer a deeper analysis or any direct comparisons. A much more complete listing of approaches for enhancing Web service performance can be found in [157]. However, that work is more general, it does not focus on mobility, and it does not use the aspects of mobile computing as criteria for its analyses. Thus, the main contribution of the survey at hand is the analysis of the adaptation mechanisms according to the "mobile computing-related" circumstances that make them beneficial, the identification and description of categories, and the mapping of related work and specific approaches to the identified categories. Some validating experiments can be considered as further contributions.

Table 3: Aspects that comprise the technical setting of mobile Web service calls and thus determine the suitability (or not) of the adaptation mechanisms. The last column indicates the thresholds used during the survey for characterizing the aspect values as small (s), medium (m), or high (h)

| COMPONENT | ASPECT | VALUES / THRESHOLDS |
|---|---|---|
| Network | Bandwidth | s: $\leqslant$ 50 kbit/s<br>m: $\leqslant$ 1 Mbit/s<br>h: $>$ 1 Mbit/s |
| | Latency | s: $\leqslant$ 10 ms<br>m: $\leqslant$ 50 ms<br>h: $>$ 50 ms |
| | Packet loss | s: $\leqslant$ 5%<br>m: $\leqslant$ 15%<br>h: $>$ 15% |
| | Stability | s, m, and h based on subjective statements of the examined publications |
| Device | CPU power | s: sensor mote or similar<br>m: phone, smartphone<br>h: laptop or similar |
| Service | Data size / SOAP size | s, m, and h based on subjective statements of the examined publications |
| | SOAP message size | s: $\leqslant$ 100 bytes<br>m: $\leqslant$ 1 kB<br>h: $>$ 1 kB |
| | Processing time | s: $\leqslant$ 1 s<br>m: $\leqslant$ 1 min<br>h: $>$ 1 min |
| Application | Service call frequency | s: $\leqslant$ 0.1 calls/min<br>m: $\leqslant$ 10 calls/min<br>h: $>$ 10 calls/min |
| | Service call criticality | s, m, and h based on subjective statements of the examined publications |

### 4.4.1 *Determining Aspects*

The consumption of Web services by wireless devices has many aspects that affect performance. After a careful study of the examined related work, four components that make up the technical setting of a wireless Web service invocation have been identified. As already mentioned in Subsection 4.3.2, these are the service itself, the device, the used network connection, and the application that uses the service. Each of

the components includes various relevant aspects, i.e., aspects that determine which adaptation mechanism would perform better under specific circumstances.

The four components, their subordinate aspects, as well as the corresponding values and thresholds used for the analysis, are listed in Table 3. For this analysis, the three discrete categorical values small (s), medium (m) and high (h) have been used for each aspect. They were considered to be a good fit because of two main reasons: Firstly, they have an appropriate granularity, which allows for an analysis of the approaches. Related work does not always provide the details that would be needed in order to support a more precise analysis. Secondly, the exact values and the thresholds change with technological advances and may also depend on the goals of the developer or the platform that want to exploit this knowledge. However, the technological advances are analogous to the workloads [17], as well as to the user expectations, so that a more coarse analysis with categorical values has much better chances of remaining valid. For some aspects, no thresholds are given at all. For these aspects, relevant statements or subjective –but logical– arguments from the related literature are used in order to fill the analysis table. The exact thresholds that are indeed provided in Table 3 are provided in order to give a feeling about the meaning of s, m, and h in the context of current applications.

### 4.4.2 *Analysis and Categorization*

As a matter of course, not every approach that could implicitly dictate a possible adaptation mechanism could be included in the survey. The approaches that have been studied and listed satisfy the following criteria:

- They are prominently published and they focus on (or refer to) mobile computing, i.e., they include an evaluation or a discussion of the benefits of the approach for wireless consumers.
- It is obvious that they can be implemented as adaptation mechanisms[2] even if the original publications do not describe them as such, i.e., they can be enforced by a mediation layer for external services without changes on the provider-side.

The studied approaches are listed in Table 4. Each approach is described by a Web service usage setting in which its authors/developers expect the approach to have a big benefit compared to the standard Web service usage. Aspects whose influence is unknown or irrelevant to the efficiency of the approach are simply omitted (indicated by the symbol "–" in Table 4). The following paragraph provides an example of how each row of the table has been filled:

In [116] and [70], for example (cf. first row of Table 4), SOAP-over-UDP (SoU) is compared to classic SOAP transport (over HTTP/TCP or directly over TCP). The transmission overhead of the three methods is compared, showing that SoU has the

2 Other approaches for lightweight remote procedure calls, e.g., the approach presented in [121], could also be interesting, but cannot be included in the study because the ability to translate their message representations to SOAP messages (and the other way round) is not proved, while the target domain and the evaluation focus are also different than the ones examined here.

3 Again here, s(=small), m(=medium), and h(=high) are ordered categorical values with $s < m < h$, so that, for example $\geqslant$ m means "m or h". As explained in more detail in the text, "–" refers to values that are either unknown or unimportant.

Table 4: Analysis of the Web service adaptation mechanisms according to the technical settings that maximize their benefit[3]

| Approach | Bandwidth | Latency | Packet loss | Stability | CPU power | Data size / SOAP size | SOAP message size | Processing time | Service call frequency | Service call criticality | Expected response time Improvement over SOAP/HTTP/TCP |
|---|---|---|---|---|---|---|---|---|---|---|---|
| SOAP-over-UDP [70, 116] | – | ⩾m | ⩽s | – | – | – | ⩽m | ⩽s | – | ⩽s | 8–10x |
| SOAP-over-SCTP [143, 165] | – | – | ⩾m | ⩾h | – | – | – | – | ⩾h | – | 1.1–1.3x |
| Compression [59, 160] | ⩽s | – | – | – | ⩾m | – | ⩾m | – | – | – | 1–1.5x |
| SOAP-over-WAP [42] | ⩽m | – | – | – | – | – | – | – | – | – | 1.3x |
| Persistent connection [59] | ⩾m | – | – | ⩾h | – | – | ⩾m | – | ⩾h | – | 2–5x |
| SOAP-over-SMTP [16] | – | – | – | ⩽m | – | – | – | ⩾h | – | – | – |
| Wireless SOAP [7] | ⩽m | – | – | – | – | ⩽s | ⩾m | – | – | – | 3–5x |
| JAVA RMI [26] | ⩾m | ⩾m | ⩽s | – | – | – | – | – | – | – | >10x |
| HHFR [97] | ⩽m | – | – | – | – | ⩽m | – | – | ⩾m | – | 1.5–10x |
| MundoCore [3] | – | – | – | – | ⩽m | – | – | – | – | – | 3–5x |
| Fast Web Services [130] | ⩽m | – | – | – | – | ⩽m | ⩾m | – | – | – | 2–10x |

smallest overhead. The overhead to user data ratio improvement is highest for small message sizes, while it gets very small with high message sizes. Therefore, as can be checked in the related evaluation results [70, 116], significant improvements are achieved with small and medium message sizes, as indicated by the notation $\leqslant$ m in Table 4. Furthermore, the response times are tested through a loopback link (i.e., a local call) and over a WLAN link. On the loopback link, the SoU improvement is small, whereas it is ten times higher over the WLAN link, which introduced a significant amount of latency. However, the UDP protocol does not guarantee a failure-free data transmission. Packets can get lost or duplicated. With packet loss values that are not small, the approach may be considered to be useless. Further, the application must be

able to cope with this feature, which means that the approach should not be used for critical service calls, but rather when their criticality for the application is small. Finally, for UDP, a timeout is defined, which is usually approximately 60 seconds. Thus, if the processing time of the service is not small or medium, there is a danger that all packets get timed out, leading to an unsuccessful service call.

A similar logic, easily comprehensible and traceable through a careful study of the approaches, was followed in order to analyze all the remaining approaches. Thus, no detailed explanation is provided for the rest of them. Instead, only the idea of each approach is summarized in one sentence: [165] suggests using Stream Control Transmission Protocol (SCTP) as the protocol for the transmission of the SOAP messages, [160] and [59] intend different types of compression for the SOAP messages, [42] uses the Wireless Application Protocol (WAP) suite and WAP gateways instead of sending the SOAP messages over HTTP, [59] evaluates the use of persistent connections for the Web service communication, [16] transmits SOAP messages with the e-mail protocol Simple Mail Transfer Protocol (SMTP), [7] introduces a more lightweight encoding technique for the SOAP messages, [26] evaluates the use of the traditional Java-based technique Remote Method Invocation (RMI) as an alternative of SOAP, [97] uses a framework called Handheld Flexible Representation (HHFR) that helps avoid the transmission of redundant data, [3] presents the *MundoCore* middleware, which is also based on a new lightweight encoding for remote calls that can be used alternatively to SOAP, while [130] describes an approach of Sun Microsystems (called *Fast Web Services*) for the optimization of the representation of Web service messages.

In addition to indicating the values that the various aspects should have so that each approach can enhance the performance, the expected performance enhancement is provided. The last column of Table 4 shows the Web service response time enhancement measured or calculated by the authors/developers of the approach. This was expressed in comparison to standard Web service calls (SOAP-over-HTTP-over-TCP) and indicates how faster the alternative solution should be in a best-case scenario. It is very important to take into account that these improvements have not been measured under counterbalanced experimental settings and have not been validated for the purposes of this survey. Therefore, they should *not* be used for direct comparisons, but rather as a reference or just in order to get a feeling of the potentials of the approaches. More important than the exact improvements are the settings that let each approach achieve its maximum benefit, because these settings can help determine which proxy would be suitable for which contexts.

Nevertheless, the results of the survey are extensible and adjustable to different requirements. For example, further approaches could be examined, evaluated, and added, gaps in the table could be filled in with values that are proven by new experiments, or existing values can be adjusted by system designers that need to use different thresholds. Thus, Table 4 is considered to form an extensible basis for decisions and further analyses and not to comprise a final and undisputable comparison result.

For a mediation layer that needs to include adaptation mechanisms which correspond to the listed approaches, it may be desirable to avoid the complexity of implementing every possible solution. Instead, only some of them may be implemented. For this reason, a further step of the presented analysis could be the identification of similarities and the categorization (or grouping) of the approaches. For example, a

mediation layer may want to be equipped with only one adaptation mechanism from each category.

At first sight, no strict categorization seems to be possible based only on the values of the aspects. Not only because these values have a coarse granularity, but also because, as mentioned, some of them are based on educated guesses rather than experiments and many entries are still blank, a strict categorization has been avoided. Instead, five non-deterministic categories are identified, which can be obtained by comparing the ideas behind the approaches. In this context, non-deterministic means that an approach may belong to more than one category, while different analysts could come up with a slightly different mapping of approaches to categories. Up to an extent, however, approaches of the same category are expected to offer benefits under similar circumstances. Other adaptation mechanisms could belong to one of these categories, but adaptation mechanisms that do not match to any of them could also appear. The following five distinguishable ideas were found behind the studied approaches:

- Reduction of redundancies in a stream of Web service calls, i.e., elimination of the repeated transmission of the same information [59, 97, 143, 165].
- Protocol transformation, i.e., replacement of SOAP messages with alternative, more lightweight, RPC mechanisms [3, 26].
- Reduction of the amount of data transmitted during a Web service call, while leaving the rest of the Web service stack untouched (solutions based on compression or encoding) [7, 42, 59, 130, 160].
- Introduction of a message queuing infrastructure that queues messages for retrieval when the device connection is ready for it [16].
- Ignorance of transmission failures for the performance enhancement of probably non-critical service calls [70, 116].

### 4.4.3 *Accompanying Experiments*

The survey has been accompanied by experiments that had two main purposes: Firstly, the conducted experiments validate some of the performance improvement expectations found in the studied literature and provided in Table 4. Secondly, they investigate if other aspects, whose influence has not yet been sufficiently examined and analyzed in related work, could have an influence on the benefit of the adaptation mechanisms, as well.

In the following, an exemplary accompanying experiment is described, which partially validates the improvements of [26][4] and [160]. Additionally, it proves that the service-related aspect "used data types" could also affect the efficiency of the mechanisms. As the influence of this aspect has never been examined in this context, this experiment is considered to be novel. However, due to the absence of relevant experiments and discussions in the related work, the aspect has not been included in the analysis of Subsection 4.4.2.

4 This has also been the only analyzed approach that is not explicitly evaluated in a mobile computing context in its referenced publication. However, the proposed technique is often referenced in this context, so that it has been interesting to include it in the accompanying experiments.

The experimental setup for the examination of the "used data types" as an aspect that could affect the efficiency of the adaptation mechanisms has been as follows:

- Two Web services have been tested. One of them sends responses with complex types (a list of complex objects), while the other sends responses with simple types (a string of varying size).
- The size of the data in the response messages has been varied from 1 to $10^6$ bytes (X-axis of Figure 12 and Figure 13). In the case of the complex data, the minimum size was approximately 2000 bytes (= size of one complex object).
- The two services were called directly with SOAP communication, as well as with the two alternative access methods, i.e., with the RMI protocol and with compression.
- The reduction of the amount of data that has been exchanged for the transmission of the responses was measured in all cases and was expressed as the size of the *adapted* response (compressed or RMI-encoded) divided by the size of the original SOAP response. This ratio is represented on the Y-axis of Figure 12 and Figure 13.

As the results in Figure 12 validate, the RMI protocol-transformation approach offers almost no benefit for services that use big-sized, simple-typed data (e.g., large texts/strings or binary data/images) and a bigger, varying benefit in other cases. The compression-based approach (cf. Figure 13) leads to different curves, offering bigger benefits in cases where the RMI protocol-transformation does not, and vice versa. More concretely, the benefit of compression increases with increasing response size, while the data type has a smaller influence in this case.

The reasons for these results relate to the type of overhead that is reduced by each of the two approaches. RMI eliminates the self-description overhead but does not reduce the size of the actual parameter values. Compression simply reduces the size of all message parts, i.e., not only of the metadata for self-description but also of the content itself. Thus, when the size of the content is dominating over the size of the metadata, compression performs naturally better than SOAP-to-RMI transformation. This happens in the case of big responses with simple data types, because complex data types need proportionally lots of metadata in the Web service response, no matter if the objects are small or big. On the other hand, SOAP-to-RMI transformation achieves a very good, i.e., low, ratio when the content of the responses is trivial. In that case, RMI messages are also trivial, while compressed messages still include the metadata, which have a considerable size even in their compressed form.

The examination of the compression approach can be even more fine-tuned. For example, parallel to the nature of the data types (i.e., simple vs. complex), the nature of the texts used in the Web service responses could be varied, as well. This is a standard factor that can affect the efficiency of compression. If auto-generated texts are used in the metadata or in the content parts of type "string", the results are usually better than when natural language is being used. However, such an analysis is here out of scope, because it does not concern other adaptation approaches and because the mentioned factor is usually unknown or cannot be influenced.

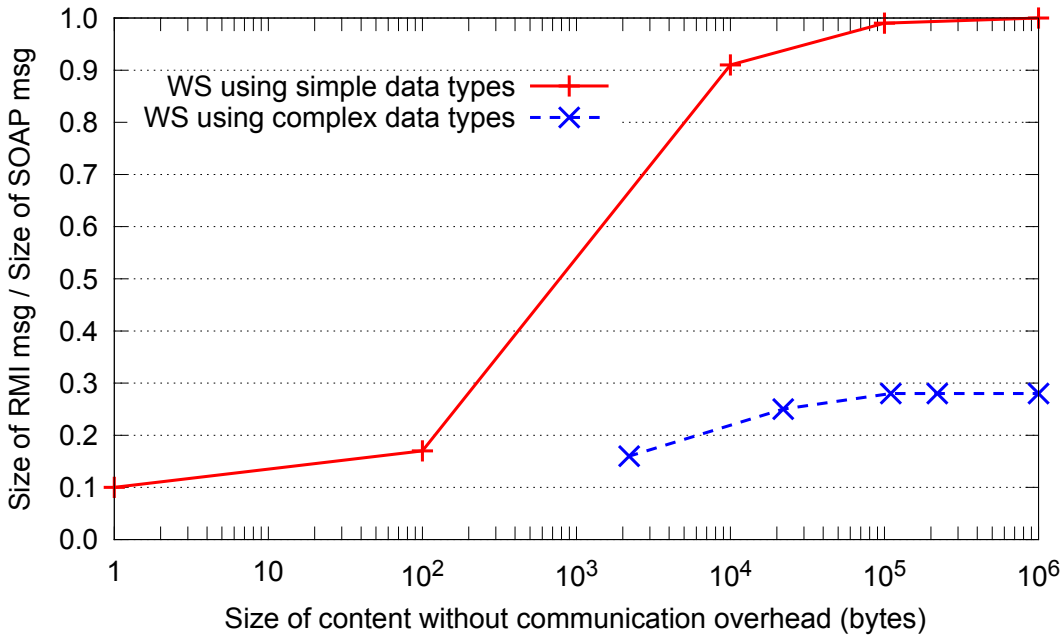



Figure 12: The effect of the used data types on the overhead reduction achieved by SOAP-to-RMI protocol translation

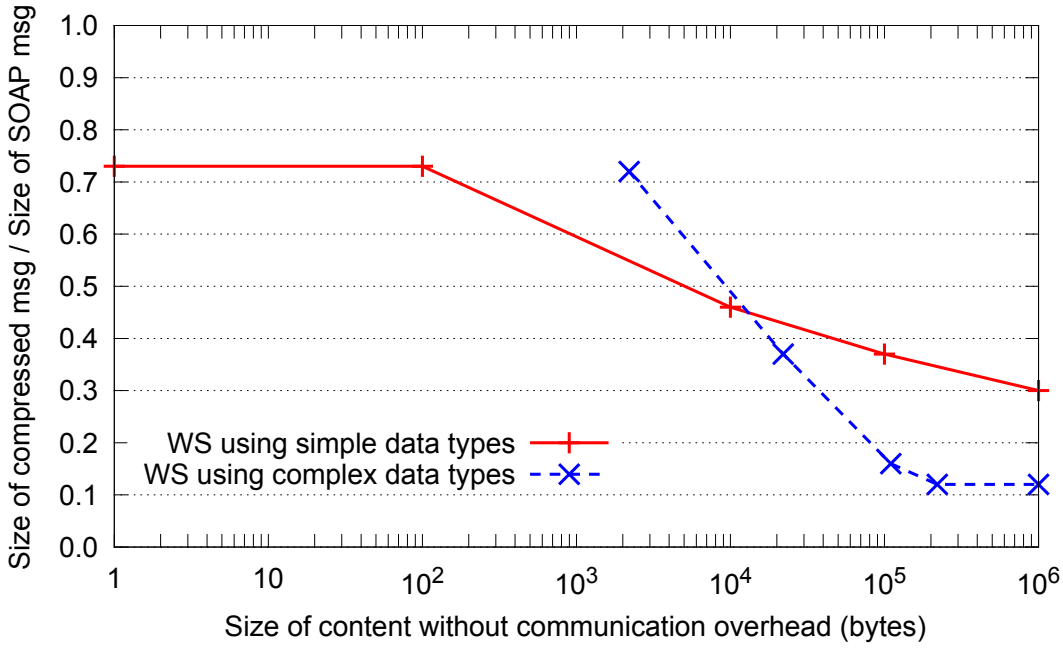



Figure 13: The effect of the used data types on the overhead reduction achieved by SOAP compression

Thus, the results have shown that, if further experimental works appear in the domain, it is always possible to extend Table 4, in order to include more details or to consider the newest technological advances. However, Table 4 is a consistent survey result, which is currently sufficient for further analyses, e.g., the identification of open issues, as will be discussed in the following Section.

## 4.5 IDENTIFICATION OF OPEN ISSUES

The presented survey has been carried out during the first steps of the MML development and its results have been carefully studied in order to identify open issues, where research questions possibly appear. Two obvious and intuitive questions are the following:

- *Can other adaptation mechanisms be developed which would be preferred under certain circumstances?*
- *Which of the adaptation mechanisms should be used in a mediation layer, for which services, and when?*

With these questions as a starting point, the study of the survey results leads a researcher to two respective, more concrete issues that are open for research:

- *Lack of caching-based adaptation mechanisms:* Caching is according to [80] one of the three main classes of solutions for mobile Web service consumption in general. The remaining two are data compression and protocol-transformation. While most of the analyzed adaptation mechanisms belong to the two latter classes, none of them is caching-based. Despite the fact that much related work has appeared in the domain "mobile Web service response caching", the proposed approaches can normally not be seen as adaptation mechanisms which could be enforced by a mediation layer for external Web services, because they usually either eliminate the Web service invocation or they need modifications at the provider-side. Even if they could be modified in order to be implemented as adaptation mechanisms, there are further challenges [159], such as the danger of using outdated information, which often prevent their use. All this is explored in much more detail in Chapter 5, which presents a new caching-based adaptation mechanism.
- *Need for decision support:* The coarse granularity of the results of the survey and the diversity of the characteristics of the approaches prove the difficulty of deterministically answering the second question stated above. Corresponding decisions are made even more complex if the mediation layer cannot or does not want to generate (or activate) an infinite number of proxies that enable the adaptation mechanisms. Therefore, the second open research issue is the investigation of the methods that can be used in order to build a decision support system that matches the peculiarities of the Web service mediation scenario. With respect to this open issue, Chapter 6 formulates an exact decision problem, describes decision support algorithms that are based on the results of the survey, and evaluates different approaches for handling the issue of missing data in this scenario.

## 4.6 SUMMARY AND CONCLUSIONS

As the focus of this thesis lies on performance-related Web service adaptation mechanisms for wireless participants, this chapter has presented the scenario in which these adaptation mechanisms shall be useful in future service-oriented systems. Further,

the MML, described in Section 4.2, has been described as a corresponding technical framework, which has been developed in order to host the Web service adaptation mechanisms.

By analyzing the scenario (IoS) and a possible technical framework (MML), it has been possible to further specify some details of the adaptation that can be performed for external Web services. This has led to the design of a case-specific, revisited adaptation loop. Among others, this adaptation loop has indicated that the relevant context information includes network-related, device-related, service-related, and application-related aspects. Further, it has pointed to the fact that the adaptation actions of interest refer to the generation of Web service proxies that enforce different adaptation mechanisms.

In order to find out which are the possible adaptation mechanisms, a related survey has been performed. The survey has analyzed and categorized a list of adaptation mechanisms that are retrieved from related publications and has concluded that most of them are based on compression, encoding- and protocol-transformations, or elimination of redundancies of the Web service messages. The results of the survey have also indicated the mobile computing context conditions that allow each adaptation mechanism to reach its maximum benefit. Thus, these results can serve as a basis for the identification of open research issues (such as the development of automated caching-based mechanisms, cf. Section 4.5), as well as a basis for decision support algorithms of a Web service mediation layer, though the latter issue is examined in Chapter 6.

# 5

# A NEW CACHING-BASED WEB SERVICE ADAPTATION MECHANISM

> »Don't worry about people stealing an idea.
> If it's original, you will have to ram it down their throats.«
>
> — Howard H. Aiken

A new Web service adaptation mechanism is presented in this chapter. The approach is enabled by mediator-based support of the caching of Web service responses on wireless devices. The proposed mechanism complements the approaches analyzed in Chapter 4 and is proven to be worth employing in a mediation layer such as the MML under certain circumstances.

## 5.1 RESEARCH QUESTION AND SCOPE

The scope of the research that has been conducted towards the development of the new Web service adaptation mechanism has been already framed through the descriptions of the scenario and the technical landscape in the previous chapter. What is missing in order to have a completely defined research scope of the contribution that is presented in the current chapter is a summary of the main idea and a formulation of the research question, as well as a description of assumptions and limitations.

### *Main Idea and Research Question*

The main idea is to use proxies on a mediation layer in order to support the application of Web service response caching on wireless devices. The more specific goal of this part is to enable the use of cached Web service responses with guaranteed 100% freshness generically, i.e., for any external Web service. As will be explained in Section 5.2, no current approaches can achieve this goal. *Design decisions* that have been made in order to achieve the desired results will be discussed along with the description of the solution (Section 5.3), whose core parts have been first published in [104]. As the mentioned enablement is pursued with the final goal of enhancing performance, the research question has to be formulated as follows:

*Can the caching of Web service responses on wireless devices be supported by a mediation layer in a way that, firstly, enhances performance compared to other approaches that achieve 100% freshness and, secondly, presents significant qualitative advantages compared to approaches that perform better by risking freshness?*

### *Assumptions and Limitations*

General assumptions and limitations refer to conditions of the future of the examined technological and scientific domains, while the more specific ones refer to scenario

and application details. Starting from the more general, and getting down to the more specific ones, the following assumptions and limitations hold for the presented solution (and its evaluation):

- *Access networks of different capabilities continue to exist:* Proxies that try to relieve the data exchange over a particular wireless part of a communication channel are meaningful only as long as access networks with different capabilities exist. However, it is very unlikely that the capabilities of all (wired and wireless) networks ever converge. For a relevant argumentation, refer back to Section 1.1, but also to the huge variety of works (of many different domains) that focus on "adaptation for mobility" (cf., for example, all chapters of [73]).

- *Self-description and verbosity remain a characteristic of SOC technologies:* The ability to achieve significantly better-performing Web service communication through adaptation mechanisms will be there as long as SOC messaging remains metadata-based and thus more verbose than its lightweight alternatives. This is also unlikely to change, because self-description and verbosity are for good reasons core features of all established SOC technologies (cf. Section 2.1). Furthermore, the Web service technology as it is established today is probably here to stay, as indicated by its very widespread usage [75], but also by its involvement in modern technologies, such as the Cloud [9, 167]. However, even if the details of the proposed solution will be tailored to the Web service technology, the main idea and many parts of the solution would be applicable for any similar self-descriptive SOC technologies.

- *Mediation layers are trusted and used (or operated by the mobile application developers themselves):* The work assumes that mediation layers such as the MML are used by mobile application developers. Although security is not closely examined in this thesis, good SOC-related security mechanisms are a prerequisite for this to happen [84], although certain security mechanisms, such as anonymity, often reduce the performance of Web service invocations [86]. Chapter 4 has explained why this is likely to happen in the IoS scenario, while the internal usage of Web service mediation layers from mobile application developers themselves is a technique that is already often employed [2].

- *Web service usage patterns that have been observed and/or predicted for current and future mobile applications remain realistic:* The evaluation of the proposed solution has to be based on realistic mobile Web service usage patterns/scenarios, which are in turn based on current applications and recent future-oriented surveys. Thus, the corresponding results are relevant as long as similar scenarios are likely to appear in real-world mobile applications.

## 5.2 RELATED WORK

Caching is only one of many approaches towards lightweight Web service consumption. When caching approaches are developed in this context and with the goal of reducing the amount of data that is being processed and transmitted during mobile Web service usage, they have to be considered in the general fields of "Web service performance enhancement" or "lightweight Web services". Indeed, this is where

the work of this chapter is positioned. It aims at achieving lightweight Web service communication based on caching. However, research for caching can be applied with small changes in many fields, as explained in detail in Subsection 2.3.2. Thus, related approaches for caching in other domains must be studied and cannot be ignored. On the other hand, specialized approaches for Web service response caching have also appeared, bringing some specific challenges and particularities of Web service response caching to the surface. In the following, our approach is first positioned in the research field where it originates from, namely the approaches for achieving more lightweight Web service consumption. Then, two classes of caching-related research works (general-purposed and SOA-related) are discussed, leading us to the identification of the gap that has not been successfully bridged yet.

*Positioning the Approach*

A listing of the most important approaches that have been designed to reduce the amount of data processed and/or wirelessly transmitted during Web service calls has been already provided in Section 4.4. Some insights of that survey, as well as insights of a more recent survey of SOAP processing enhancements [157], are briefly summarized here in order to explain why it has been decided to pursue the use of caching as a Web service adaptation mechanism for lightweight Web service consumption.

Most of the adaptation mechanisms examined in Section 4.4 fell into one of the following categories: reduction of redundancies, on-the-fly protocol transformation (SOAP-to-X), and compression (or optimization of the representation). Obviously, the use of caching in the form of an adaptation mechanism would be a category by itself. A proof of this is that all the approaches listed in Section 4.4 had calculable upper limits for the "message size reduction ratio" that they can achieve. Such limits do not exist for an adaptation mechanism that is based on caching, because the mentioned ratio is case-dependent and can, theoretically, have very big values, in cases where the complete content of the response does not have to be transmitted.

The survey of [157] does not focus on mobile computing and is not limited to the examination of possible adaptation mechanisms, but it rather considers all SOAP performance enhancement approaches. An important consequence is that some of the approaches described there may not be applicable without changes of the original Web service, i.e., may not make much sense when external, third-party services are considered. However, this survey helps position the current work more precisely, because it defines six levels (i.e., six points/phases of the "lifecycle" of a Web service call), where the focus of the enhancement may lie. These are the following: serialization, parsing, de-serialization, security policy evaluation, compression, and multicasting. Although approaches that are limitless in terms of the "message size reduction ratio" may be included in this survey (a corresponding analysis is not provided), it is self-evident that enhancement approaches with different focus are beneficial under different circumstances, depending on the goals of the system. For example, a client-side caching approach is listed as an enhancement of the serialization phase, because it does not always reconstruct identical requests. However, if we do not consider caching as a client-side optimization but as an adaptation mechanism of the service itself, i.e., if the idea of caching is not used in order to avoid connection

establishments, but in order to reduce the size of messages (as the presented approach will do), then the corresponding approach that falls under a different category, creates new potentials, and achieves different goals.

*Caching In General*

Much of the research effort in the area of caching has been devoted to the development of cache replacement strategies [117]. Such strategies concern the maintenance of the cache and their goal is to retain in the cache the entries that are most likely to be needed again soon, i.e., to maximize the cache hit ratio. Although most of the caching approaches are based to some extent on the classical strategies Least Recently Used (LRU) and Least Frequently Used (LFU) [60]), the research interest in the field is still alive. New approaches are still being developed, exploiting the characteristics of particular technologies in order to be more efficient. For example, [53] recently developed a replacement policy that has better performance than LRU and less complexity than LFU in the case of Zipfian request probabilities and big cache sizes, while [18] has enhanced cache management for mobile computing systems based on a concept for adaptively prefetching the content.

However, enhancing the hit ratio does not enhance at all the freshness of the cache information. Thus, if the usage of up-to-date information is of high importance, additional techniques have to be used in order to update the cache regularly. While the aforementioned solutions focus on the cache hit ratio, the state-of-the-art solution to optimize the cache freshness is the use of invalidation reports [19]. In that case, the servers indicate the changed data items to the clients at intelligently-determined intervals. Otherwise, the freshness normally just relies on "Client Validation", which means either "polling every time", or defining an adequate TTL for the cached objects, as is explained in more detail in the survey of [20].

Either because of their scope (i.e., they often do not consider freshness at all) or because it is impossible to apply them "as they are" in an arbitrary domain, these general-purposed works may be very important and fundamental, but they are not always sufficient when it comes to the caching of Web service responses. This is explicitly argued in related surveys and implicitly proven by the fact that specialized approaches for Web service response caching are intensively researched.

Supporting the aforementioned argument, Table 5 summarizes some important differences between Web content (WWW) caching and Web service response caching, as they are, implicitly or explicitly, found in related sources (e.g., [43, 74, 135, 159]). The implications that these differences have, or the challenges that they generate, are also discussed. It must be noted that Web content caching is most closely related to Web service response caching than any other type of caching, because of their technical similarities, e.g., the fact that they are both handled by Web servers, they are (normally) both transferred over HTTP, they use similar content formatting (HTML vs. XML) etc. Among the differences discussed in Table 5, most interesting for the work at hand are the ones related to the cache (lookup) mechanisms, because they comprise the main reason why the details of the solution presented in Section 5.3 are irrelevant for any related work in the domain of Web content caching.

Table 5: Differences between Web content (WWW) caching and Web service response caching summarized from [43, 74, 135, 159]

| ASPECT | DIFFERENCE | IMPLICATION/CHALLENGE |
|---|---|---|
| Cache (lookup) mechanisms | The content of Web service responses is hidden in a document part which is not handled directly by the Web server | Contrary to Web content, HTTP header details cannot be used as (caching) keys for Web service responses |
| | The same contents are very often enclosed in (slightly) different documents | Identifying cache hits or checking consistency is not as easy as checking if two documents (or document parts) are identical |
| | Web service responses are not stored in an hierarchical structure of Web proxies, which is often an important characteristic of Web content storage | Caching approaches that are oriented to or dictated by this classic proxy structure are irrelevant for Web service responses |
| Content features | Web services are often part of known (or discoverable) processes, which are sometimes even formally described | Cache maintenance, or even prefetching of responses, can be based on the knowledge about these processes |
| | The diversity of Web service operations makes it difficult to sort them as "read" or "write" actions | Keeping cache consistency by checking simple dependencies and performing simple updates is not straightforward |
| Security | Web services use authentication mechanisms more often and in a different way than web sites | Cache managers usually handle only content, but they should perform the necessary checks if authentication is demanded by the providers |

*Caching Web Service Responses*

By studying the particularities of Web service response caching (i.e., questions like: what are the technical differences compared to caching in other domains? What phases of Web service calls are the most resource-consuming? What are the alternatives for the representation of cache responses?), some researchers came up with specialized solutions for the field. Before proceeding to the description of proposed solutions, [159] is referenced as one of the most detailed analyses of the peculiarities of Web service response caching for mobility. A demand for new technical solutions and standards, rather than algorithmic extensions, has been identified, because most of the challenges were related to technical enablements, and not to enhancements of algorithmic efficiency. Thus, implicit directives such as those provided by [159] led researchers to suggest three different types of solutions: *provider-based*, *mediator-based*, and *client-based*. The categorization is not perfectly unambiguous, but it depends on

where the main components or the important caching logic of each approach are located and it helps to identify aspects that each class of solutions cannot address:

*Provider-based* solutions rely upon logic or modules that are added to the host of the original Web service. Thus, they have the major drawback that they require modifications of the original service (or of its hosting system). Because of the target scenario of this thesis, such solutions come out of question. Any proposed solution should be able to work with existing (probably third-party) services without "touching" them. However, some techniques and ideas that appear in provider-based solutions may be useful. [79] describes an approach based on metadata that indicate if Web services are cacheable or not. The important issue of cache performance vs. cache freshness (or consistency, as it is called there) is mentioned, but without any attempt to maximize the two properties at the same time. Further, the authors focus on the case of "connectivity loss", which is not focus of this thesis, while they also rely upon provider-side metadata and modules. In [74], the provider responds with a hashcode instead of retransmitting an identical complete response, when a client requests unchanged data. Unfortunately, it is not clear if and how the provider-side middleware (called SigsitAccelerator Proxy) could work outside the provider system. Furthermore, the proxy might become a bottleneck, because it seems as if there is only one dedicated proxy for all backend services. Even if the above problems could be solved, a proof that the proxy could be generated for any Web service and a technical description of how this could be achieved in an automated manner would be still missing. Further, the solution of [74] is not designed for and evaluated in the context of mobile computing. Some interesting ideas about efficient caching based on metadata-based knowledge about how response messages are generated is presented in [154], but this work is, again, strongly dependent on a new architecture and new standards for the provider, while the freshness of Web service responses is not considered at all.

*Mediator-based* solutions need modules that are hosted separately from both the provider and the consumer. Although many of the referenced works mention the use of proxies, they are not considered here as mediator-based if the proxy is actually part of either the provider- or the client-system or network (which is often the case). In order to consider an approach as mediator-based, it must be possible (and it must also make sense) to physically separate the mediator (or proxy) from both the provider- and the client-system. [135] presents an efficient mediator-based technical solution about how mobile Web service response caching should be handled when processes are concerned. Processes can be learned and responses can be prefetched. Nevertheless, prefetching is obviously forbidden when the goal is 100% freshness. CRISP [34] can also be used as a mediator-based solution, because its architecture enables method-call interception without further requirements at the provider-side. However, the existence of two modes ("consistency mode" and "performance mode") proves that the constraint mentioned above for [79] is present in this work, as well. The approach presented in this chapter is mediator-based but it differs from the referenced approaches in that the proxies at the mediator-side are generated automatically and separately for each service and, of course, in that it will guarantee 100% freshness.

*Client-based* solutions enrich only the client-side (usually with some client-side caching middleware) and do not need any other entity elsewhere. They attempt to enhance the way a client can store, represent, handle, replace, and re-use identical requests and responses. However, when a message-exchange does occur, it conforms exactly to the message-exchange that occurs in the absence of any caching mecha-

nism. Within the context of this thesis, [152] presents the most relevant client-based approach. There, the idea of using "browser-like" responses that replace the content with freshness codes (cf. Subsection 5.3.1) is theoretically mentioned, but it is not further handled and it is only mentioned as a technique for the representation of responses that come from the client cache and thus not as a technique that can be used for reducing the wirelessly transmitted data. The idea was not considered from a technological point of view and it was, of course, not generically enabled for existing Web services. The work assumes that "it is the responsibility of the client application administrator to configure a TTL for each operation". The existence of TTL is a proof that the risk of using outdated data is accepted. A very important point of that work is the idea to compare application objects instead of XML messages. This idea will also be embraced in this chapter, but will be implemented in a more generic manner, because the modules that compare the responses will not be built once statically, but they will be built automatically for every Web service, thus being able to handle every possible application object. It cannot be judged if the cache of [152] uses some other way in order to be able to handle every application object, but many of the previously referenced approaches compare XML messages. This is slower and reduces the probability of a "match" because of small, unimportant, content-irrelevant differences that may appear in an XML message.

*Putting it all together*

All in all, Web service response caching has always been handled as a feature that can be inherent part of a Web service system or not. In none of the referenced works has caching support been examined as an adaptation mechanism for existing Web services. Further, the focus of related scientific work has rarely been on freshness. *Tables with detailed comparisons will be provided in Subsection 5.3.4*, after the approach of this thesis has been described. The latter approach will be referred to as Proxied Calls with Validity-checks (PCV).

## 5.3 AN ADAPTATION MECHANISM BASED ON RESPONSE VALIDITY-CHECKS

Having identified in Section 5.2 a particular gap, the current section presents the core idea and the details (techniques and algorithms) of a solution for bridging this gap.

### 5.3.1 *The Novel Idea of Validity-Check Enablement for Web Services*

In the following, an important constraint of Web service responses caching is introduced and described, along with an explanation of why the generic withdrawal of this constraint is just now starting to be motivated in modern service-oriented landscapes. The core idea of PCV is to be able to automatically withdraw this constraint from any Web service without having access to its hosting system.

Naturally, with or without mediation layers being involved, every client can store Web service responses for future usage. However, there are three main reasons why a client may avoid doing so:

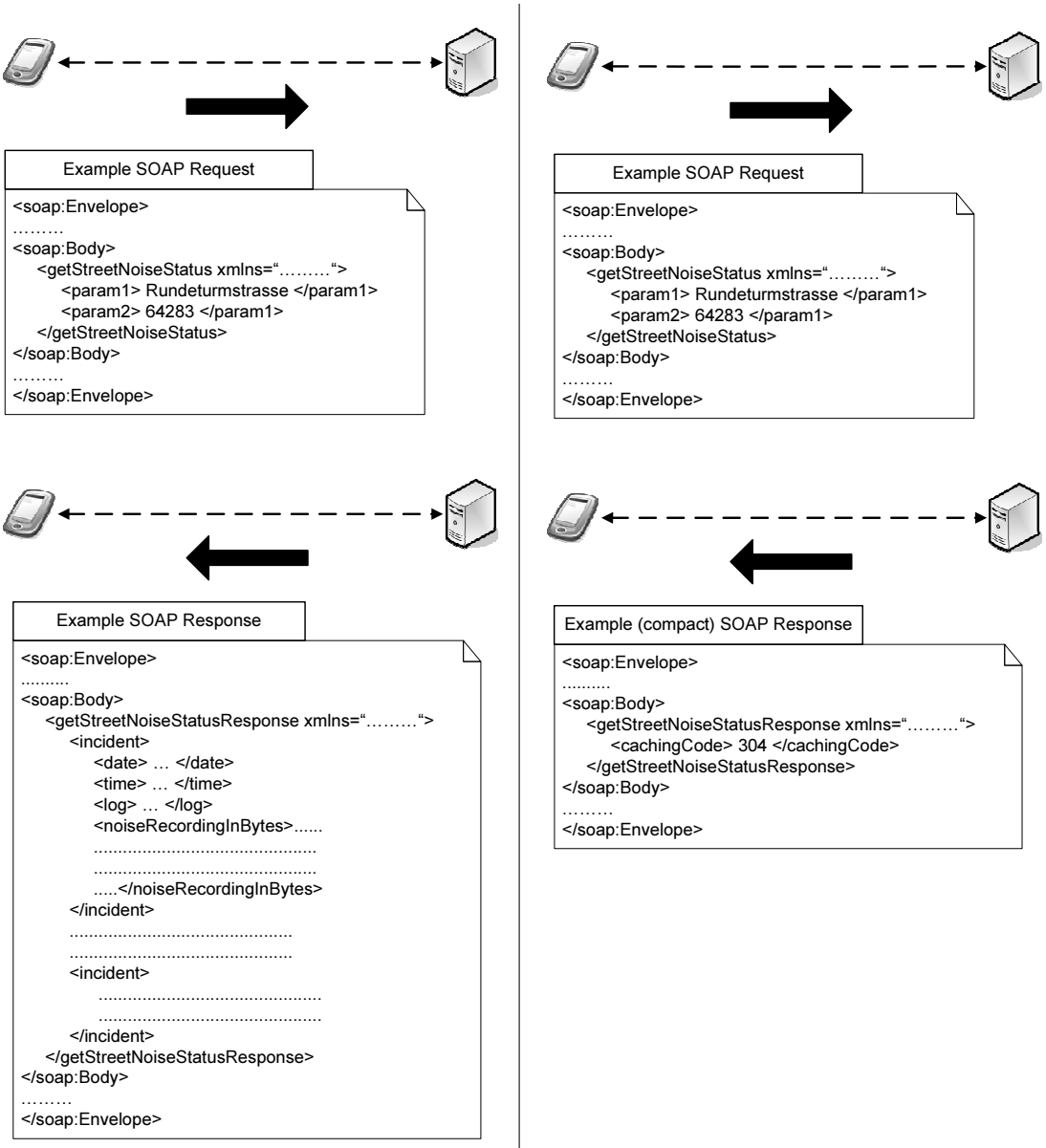
- *Frequent changes:* The content changes so frequently that client-side caching does not make any sense.
- *Criticality:* The service call is so critical for the user that she/he needs to be 100% sure about the validity of the content, even if the old response has good chances of being up-to-date.
- *Legal issues:* Caching may be implicitly or explicitly stated to be illegal for a particular service.

In a global marketplace, it is difficult or unusual to know about these features, unless relevant information is included in an extended service description. But even if it is known that one of the above statements is valid, the best that the user can do is always request a new response.

Furthermore, Web Service Response Caching presents the following important constraint: As already identified shortly after the appearance of Web service technologies, XML-based Web services present many technical challenges that prevent them from being involved in a caching process similar to the one used for simple Web content (e.g., pure HTML) [159]. Based on directives provided by Web servers and supported by Web browsers, simple Web content is not reloaded each time it is requested, if the old (cached) content is still valid. The main technical reason why this mechanism cannot be transferred to the Web service technology is that Web services do not just rely on the HTTP-GET method for their communication, but they implement a rather more complex message exchange in order to export a diverse set of operations. As a result, Web services always transmit the complete response when they receive a request. The only alternative in order to support a validity check is to manually implement it inside the logic of each service. A survey among 20 services provided at www.webservicex.net (the 10 most popular and the 10 most recent) has revealed that not a single one of them has inherently implemented such a logic. The reason is obvious: the individual implementation effort outweighs the benefits, while the latter may often concern a limited number of clients. But what about a generic solution that could add this feature to any Web service in the IoS? Why has research not focused on it until now and what exactly would it enable?

In fact, the MML (or any similar mediation layer) is one of the first systems confronted with a scenario where such a generic "browser-like" solution would be worth its effort, because:

- The MML has access to a huge amount of services over the global marketplace. Some of them would benefit from being enhanced with support for safe client-side caching. However, the MML cannot extend them directly with such logic, because it has no access to their code or their hosting systems.
- The MML target group consists exclusively of wireless clients, while the service providers are often more concerned about larger PC-based clients.
- The MML may be able to estimate better which services (or their clients) would profit more from client-side caching, because of extra information about "cacheability" that it could theoretically retrieve from USDL descriptions. However, a general-purpose mechanism should be able to function even without this extra information.




a) SOAP request / response pair with the typical communication scheme

b) SOAP request / response pair with the envisioned communication scheme

Figure 14: Visualization of the idea of "browser-like" validity checks for Web service responses

- The adaptation of Web services for mobility is the business model of the MML. Thus, the access either to the proxies generated by the MML or directly to the MML software in order to generate own proxies would be tradable goods in line with the MML business model.

For these reasons, the MML needs a mechanism which, when applied to any external Web service, allows the mobile clients to perform service calls that have the chance to be satisfied with very small, lightweight responses, whenever there is no reason to retransmit lots of data. Upon receiving responses with particular codes in their content (for example, 304, as in simple HTTP calls), clients would know that their

cached data was still up-to-date. This would not eliminate the need for establishing a connection, but could significantly reduce the wirelessly transmitted data. In any case, there is no way to be 100% sure of the validity of the data without establishing a connection each time that the data is needed. The principal difference between typical Web service communication and the communication scheme that PCV will enable is abstractly explained in the following with the help of an example.

Figure 14a shows a SOAP request/response pair of an example Web service that informs its invoker about recorded noise incidents at a particular place/address. Such a response may contain binary data, resulting in very big content sizes. This happens mostly in cases where the responses contain multimedia content, because multimedia representation often involves the use of big data volumes [149]. Indeed, it is not rare for Web services to be used in the multimedia domain [177]. Of course, this is only an example. Responses that do not contain binary data may also become very big, as will be discussed during the evaluation. Currently, with the typical implementation of Web services, the transmission of a request such as this of Figure 14a leads *inevitably* and *every time* to the transmission of a complete response.

The idea is to enable a communication scheme where it can be automatically recognized if the response contains any data that would be "new" for the invoker. In that case, instead of receiving a response such as this of Figure 14a at every invocation, the invoker would sometimes receive a response that signals the fact that the invoker still has up-to-date information from a previous invocation and can thus reuse it. An example response is shown in Figure 14b, making it obvious that there is a potential for much more lightweight communication.

Enabling request/response pairs such as those shown in Figure 14b normally requires reengineering and reprogramming of the Web service. However, the MML needs a generic solution which somehow enables the described vision for any third-party Web service without access to its code and can be used by the clients without having to bother about any of the obstacles listed previously (frequent changes, criticality, legal issues).

### 5.3.2 *Benefit Analysis of the Enabled Communication Scheme*

Figure 14 has given a first impression of how much data traffic can be saved if validity checks are enabled for the Web service communication. However, a Web service-related analysis based on response parts and appropriate caching variables is necessary in order to mathematically express the effect that the use of validity checks has on data traffic.

Assume a sequence of identical service requests and the following scenario-related variables:

- $s_{avg}$: Average response size.
- $s_{304}$: Size of a compact response (cf. Figure 14b). Note that, contrary to the size of original/complete responses, $s_{304}$ is *not* variable.
- $r$: Hit ratio. Not that in the case of validity-checks, a hit refers to a Web service call that is satisfied by a compact response, while in the case of traditional client-side caching, a hit means that the response is found in the client cache.

- $h_{benefit}$: Average amount of data spared by a hit.
- $m_{loss}$: Average amount of additional data needed in the case of a miss.

Finally, the amount of data traffic saved per service call in comparison to standard Web service communication (without caching) is used as a metric, and denoted as $b$ (for benefit).

Obviously, for the Web service call sequence enabled by *PCV*, the benefit is:

$$\begin{aligned} b &= h_{benefit} \times r - m_{loss} \times (1 - r) \\ &= h_{benefit} \times r - m_{loss} + m_{loss} \times r \\ &= (h_{benefit} + m_{loss}) \times r - m_{loss} \\ &= {}^{1}((s_{avg} - s_{304}) + s_{304}) \times r - s_{304} \end{aligned}$$

so that, finally,

$$b = s_{avg} \times r - s_{304} \tag{5.1}$$

In order to better understand what this practically means, assume the existence of an *optimal client-side caching approach*, namely of one that has all previous responses in its cache and "magically" knows when to use them, i.e., transmits a new request only when the cached response is outdated. Obviously, the hits and the misses (and, consequently, the hit ratio) coincide with those of the validity-check case examined above. Thus, the benefits can be examined based on the same terms, as they have been defined above. The benefit of this theoretical, optimal caching solution would be then:

$$b = s_{avg} \times r \tag{5.2}$$

because it would not transmit any data in the case of a hit, while $m_{loss}$ would be equal to 0. Equation 5.2 provides also a *theoretical upper bound* for approaches that guarantee the freshness of the used responses. Interestingly, a comparison of Equation 5.1 and Equation 5.2 indicates that, for *nonzero* values of $r$, the benefit of PCV converges to the theoretical upper bound for increasing values of the quotient $s_{avg}/s_{304}$.

The quotient $s_{avg}/s_{304}$ is of course unknown before the actual usage of the service. Neither can it be calculated by parsing the service description, because the descriptions cannot be aware of the exact amount of data that the users will exchange with their Web service messages. However, it is interesting to note that some coarse estimations of the quotient could be performed by analyzing the service description. For example, the amount of parameters of an operation, as well as their types and name lengths, can help calculate the size of the XML metadata needed in order to wrap the actual data in the corresponding responses. To provide a simplified example, $n$ parameters (of primitive types) appearing one after the other with an average name length of $l$ would produce in the response an overhead of $n \times (2 \times l + 5)$ characters for the

1 For this step, the following (pessimistic) assumption is made: $m_{loss}$, which is caused by the extra fields that the validity check needs to use (e.g., ifModifiedSince, cachingCode), is equal to $s_{304}$. Actually, $s_{304}$ already includes (at least) the extra fields and probably more data, so it is indeed true that $m_{loss}$ is actually always smaller or equal to $s_{304}$, as has been also confirmed by practical tests with the implementation that is described later.

metadata, according to the scheme `<parameterName>...</parameterName>`. Similar but more complex functions could calculate the overhead for the complete response, helping to estimate $s_{avg}$ and, with that, the quotient of interest.

### 5.3.3 *Automated and Generic Web Service Proxy Generation*

The described idea is enabled by a solution called Web Service Proxy Generator (WSPG). In addition to proxies for caching support, the WSPG can also be extended to generate other proxies for overhead reduction (e.g., protocol-transformation, compression). Although parts of the proxy generation process that will be presented can be used for these other kinds of proxies, as well, all descriptions that follow refer to the examined caching-support proxies, because of the innovative concepts, techniques and evaluations that are involved in their case. To get a good understanding of the (caching) WSPG, one should also refer back to its environment and its technical landscape (Figure 9 and Figure 10) and be reminded that it should generate proxies of external, third-party Web services for a mediation layer. Accordingly, the following paragraphs explain how the caching (or *validity-check enabled*) proxies are generated based solely on service descriptions. Firstly, the workflow of the proxy generation process is described on a high level and, secondly, a closer look at the workflow and at some important details of the proxy generation process are provided.

#### *The Logic of the Generic Proxy Generation*

Figure 15 shows the steps of the workflow that is used in order to achieve the automated and generic proxy generation. Each step will be described separately. The descriptions are provided from an engineering point of view, though some helpful references to the Java-based implementation are made whenever this can potentially sharpen the understanding of the reader. The highlights are to be identified in the conception of the workflow itself and in the innovative code adjustment techniques (esp. of Steps 3 and 4), but also in the difficulty of implementing such a logic generically, i.e., for any external Web service.

- *Step 1 – Description Parsing:* All the necessary information, such as operations, data types, interfaces, ports etc., is read from the service description. Because specialized information is needed, but also because the parser has to be extensible in order to support other Web service descriptions standards (in addition to WSDL), the WSPG does not use an off-the-shelf service description parser but a new one. An important scenario-related part of its logic is explained later in this section, based on an algorithmic extract.

- *Step 2 – Consumer Generation:* This step generates the part of the proxy that is responsible for calling the original Web service. As no sophisticated or scenario-related actions have to be performed during the consumer generation, the standard tool *wsimport*[2] is transparently and automatically used by the WSPG for this step.

2 `http://download.oracle.com/docs/cd/E17802_01/webservices/webservices/docs/2.0/jaxws/wsimport.html` (Last accessed in January 2012).

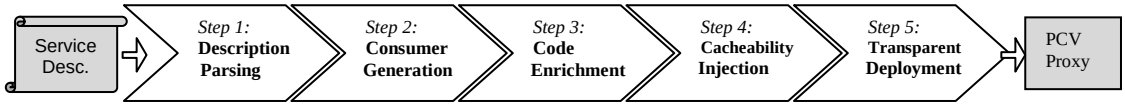



Figure 15: Workflow of the Proxy Generation Logic

- *Step 3 – Code Enrichment:* The code generated by wsimport is different for each service. Each time, the WSPG has to search deep into that code in order to find the points that have to be enriched. Then, it automatically modifies and adds code as needed. For example, (i) the addition of extra parameters in the request/response wrappers and (ii) the adjustment of the interface classes, which determine how the new WSDL (i.e., the WSDL of the proxy) will look like, are two important code enrichment actions.

  With regard to (i), a request (or response) wrapper is a class that determines what the SOAP request (or response) contains. After detecting these classes by analyzing the annotations of their code, the WSPG inserts the field *ifModifiedSince* to the request wrapper and the fields *statusCode* and *identTag* to the response wrapper. These newly inserted fields are always preceded by the annotations that are necessary for their inclusion in an XML message and accompanied by the corresponding getters and setters. This allows for a communication pattern such as the one presented in Figure 14b and for validity checks implemented based on the comparison of *ifModifiedSince* parameters with previously saved *identTag* parameters.

  As for (ii), the WSPG has to detect the interface classes and replace pairs of RequestWrapper/ResponseWrapper annotations that are included there with the annotation: `@javax.jws.soap.SOAPBinding(parameterSytle=ParameterStyle.BARE)`. This allows for the use of extra parameters (see previous paragraph) without having to change the Web service signature, i.e., without additional wrapper classes for the new service.

- *Step 4 – Cacheability Injection:* Similar code adjustments are also needed for the cacheability injection. However, this step concerns the actual handling of the cache, i.e., of the classes that access it or support it. Its logic is separate from the previous step, and most of the results of the code enrichment actions of Step 3 are preconditions for the cacheability injection.

  The most interesting actions of the WSPG during this step are the insertions of appropriate annotations to all the entities that must be persistable, i.e., are involved in the storage of request/response pairs, as well as the automatic generation of two further modules for each operation: A *builder* module and a *controller* module.

  The builder module is capable of storing request/response pairs so that validity checks can be performed. Note that the automatically generated databases that are used by the builder module are different for every proxy. This is because it is desired to use service-specific databases that store application objects, rather than generic databases that store XML documents.

  The controller module can search among the mentioned pairs and compare new responses with cached ones, in order to decide whether the new response or

just a status code should be sent back to the wireless client. Like the builder module, also the controller module includes service-specific parts.

- *Step 5 – Transparent Deployment:* In this step, not only the code that results from the execution of Step 4 has to be packed and deployed but also the file structure and the configuration files of the Web container that will host the proxy have to be transparently adjusted. After this step has successfully completed, the generated proxy becomes automatically available as a new Web service. In terms of content, the Web service description of the proxy has only trivial differences to the description of the original external Web service.

*Important Details of the Proxy Generation Process*

In order to make the proxy generation process more comprehensible and traceable, a sequence diagram is presented, along with extracts of algorithms that highlight and further explain some important procedures. The latter procedures include many of the parts that differentiate the presented approach from standard approaches. These parts will then be summarized again in Subsection 5.3.4.

The sequence diagram of Figure 16 reveals how the proxy generation program runs through concrete actions of the five workflow steps. An understanding of the workflow described in the previous subsection is necessary in order to study the sequence diagram, while algorithms that refer to the actions 1.2, 3.1, and 4.3 are provided separately with the help of pseudocodes (Pseudocode 1–Pseudocode 3). It is reminded that this sequence diagram does *not* depict how a caching proxy operates, but how the proxy generator works. The depicted entities should be understood as follows:

- The *Proxy Creator* is the module that coordinates the process and stores globally needed information. It also organizes the generated software parts, ensuring thus the easy integration of the generated proxies into the system(s) where they will have to operate. Thus, it is the entrance and the exit point of the process and it is mainly involved in the workflow steps 1 and 5.

- The *Description Parser* analyzes the service description. As shown in the sequence diagram, it is used by the Proxy Creator and, apart from integrating an XML parser, it has no further interactions with other modules. However, it includes the logic for retrieving the proxy-related information from the parsed Web service description. This logic differentiates it from existing Web service description parsers, as will be explained in the discussion of Pseudocode 1.

- The *Generation Module* is the module that actually implements all the complex tasks of the workflow steps 3 and 4. After initiating the consumer generation, it generates various classes, the most important of which are shown and explained in the diagram and in the corresponding pseudocodes.

- The *External Modules* include all third-party libraries, as well as any non-source-code-related components that are used throughout the process (compilers, tools etc.).

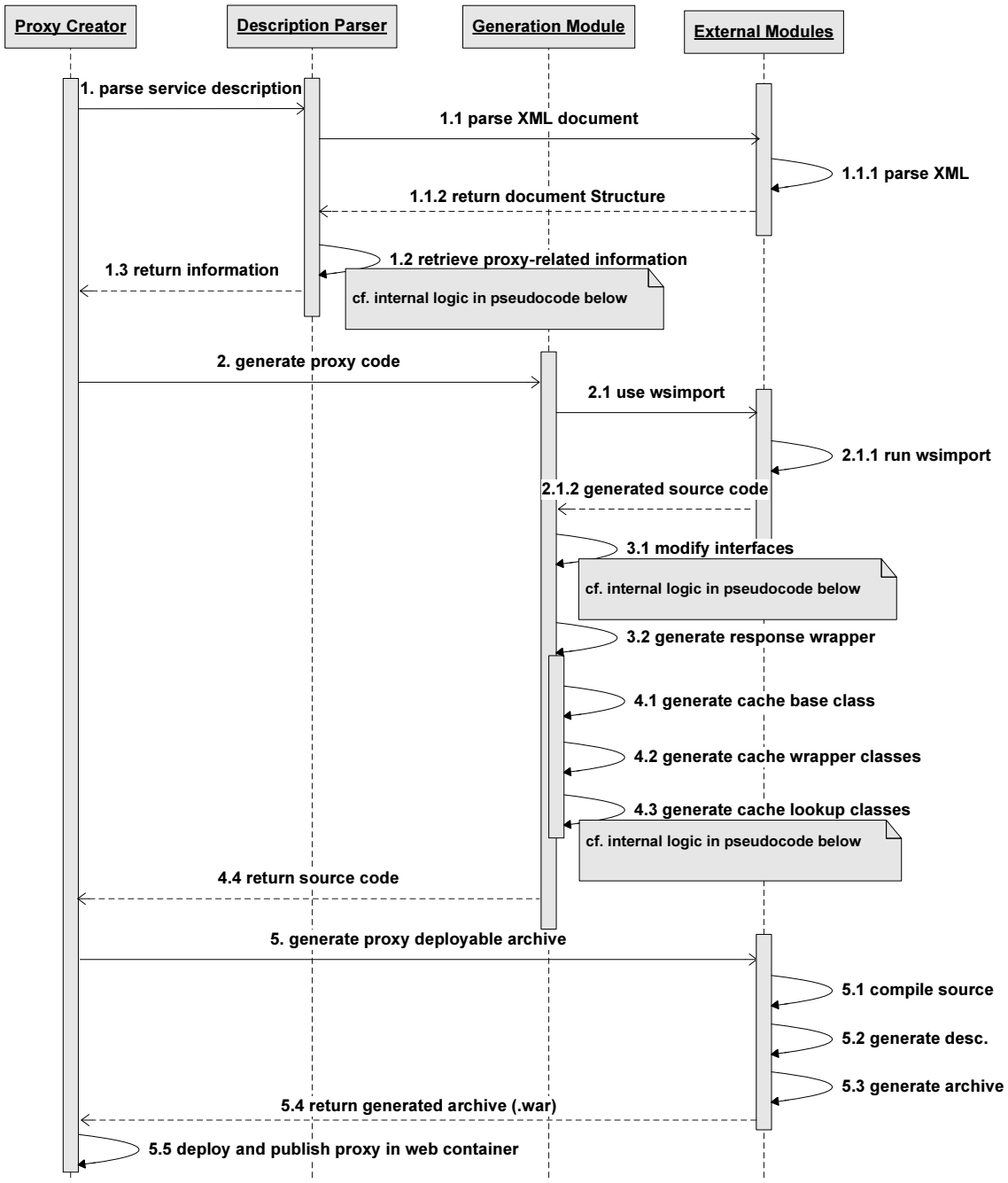



Figure 16: High-level sequence diagram with the main entities of the WSPG

Pseudocode 1: Retrieve proxy-related information

```
// Input:     descDocument (the service description document)
// Output:    service (the variable in which the service info will be stored)

// '#' means "value of" and ':' means "element of"

descService = parse_XML_description(descDocument)
if valid_description_document()
   service.name = #(descService.name)
   service.package = #(targetNamespace).toPkgName()
   for_each binding:descService
      endpoint.name = #(binding.name)
      for_each operation:binding
         params<name, type> = operation.findAllParams()
         requestWrapper = find_wrapper_pkg_and_type_in_descService()
         responseWrapper = find_wrapper_pkg_and_type_in_descService()
         endpoint.addOperation(name, #(requestWrapper.class), #(responseWrapper.
             class), params)
      end for
      service.addEndpoint(endpoint)
   end for
end if
return service
```

Pseudocode 2: Modify interfaces

```
// Input:     service (the variable in which the service info is stored)
// Output:    -

// '#' means "value of" and ':' means "element of"

src = `` #std_package + #(service.package) ''
src += `` import ......'' // import all Web service models/annotations
src += `` web_service_annotation '' // (e.g. @WebService)
src += `` class #(service.name) ''
for_each operation:service
   src += `` request_wrapper_annotation ''
   src += `` #(requestWrapper.class) '' // ...of the current operation
   src += `` response_wrapper_annotation ''
   src += `` #(responseWrapper.class) ''// ...of the current operation
   src += `` set_ParameterStyle_to_BARE() '' // to allow modified wrapper
       objects to transport additional information like ``ifModifiedSince'' etc.
   src += `` public #(responseWrapper.class) #(operation.name)() { ''
   src += `` modified_method_impl ''
   src += ``}''
end for
wirte_#src_into_new_file()
```

Pseudocode 3: Generate cache lookup classes

```
// Input:     service (the variable in which the service info is stored)
// Output:    -

// '#' means "value of" and ':' means "element of"

for_each operation:endpoint
   src = '' #std_package + #(service.package) ''
   src += '' import ......''
   src += '' class #(operation.name).toCacheName() extends CacheBaseClass ''
   src += '' List<(operation.name).toRequestWrapperName(), (operation.name).
       toResponeWrapperName()> cached; ''
   src += '' constructor() {cached <- popupate_Cache()} ''
   // requests and responses in the following contain
   // ifModifiedSince and IdentTag attributes etc.
   // NOTE: Talking about requests & responses is a further
   // abstraction in order to explain the logic of the
   // auto-generated method lookupCache.
   // Don't forget that for each operation, requests and responses are
   // objects of different data types
   src += '' lookupCache(request) { ''
   src += ''    if compare_iteratively(request, x:cached) = true ''
   src += ''       if mediator_offline_caching_allowed() ''
   src += ''          return Code_303 '' // mediator had recent response
   src += ''       else ''
   src += ''          response = call_Service(request) ''
   src += ''          if (response = response_in_cached) ''
   src += ''             return Code_304 '' // client response was fresh
   src += ''          else ''
   src += ''             add_to_cache(response) ''
   src += ''             return response ''
   src += ''          end if ''
   src += ''       end if ''
   src += ''    else ''
   src += ''       response = call_Service(request) ''
   src += ''       add_to_cache(response) ''
   src += ''       return response ''
   src += ''    end if ''
   src += '' } ''
   // The lookup class for each operation is stored separately
   Create_appropriate_package_directory(#std_package, #(service.package))
   wirte_#src_into_new_file()
end for
```

The purpose of the algorithms in the pseudocodes is to explain the internal logic of important actions, because this logic does not become obvious from the sequence diagram alone. Furthermore, some of these actions are closely related to aspects that distinguish this work from related approaches. The algorithms are high-level, as well, and the provided parts may correspond with hundreds of lines of code in the actual implementation:

- Pseudocode 1 shows how the description parser preprocesses and stores the necessary information in a way that will best support the next steps of the

workflow. It shows, for example, how the package structure that will be needed is prepared based on the target namespaces, which are found in the service description. If the packaging was not based on the target namespaces the way it is done, it could lead to the automatic generation of wrong or inconsistent code, which would not compile. The next lines gather and store the operations-related information (name, wrappers, parameters) exactly as they will be needed later for the code enrichment and the cacheability injection. Such actions are not included in state-of-the-art (WSDL) description parsers.

- The way in which the WSPG digs into the generated code in order to add the annotations and the code that will determine how the new service interface and, accordingly, the new service description (i.e., that of the proxy) will look like, is shown in Pseudocode 2. The pseudocodes indicates how wrapper annotations are added directly before the declarations of class variables. As can be seen, the types of these variables are those that have been identified by the parser (cf. Pseudocode 1).

- A closer look at the automatic generation of the class that performs the actual comparison of Web service responses (Pseudocode 3) is also interesting. Contrary to the generation of the caching *base class*, which is similar for all services/operations, the *cache lookup classes* have many operation-specific parts. This makes it very difficult for the Generation Module to perform its task generically. The pseudo-code in Pseudocode 3 summarizes (among others) how (i) the generated code must use the particular request/response wrapper objects that correspond with the target operation, (ii) a constructor must be built that is able to populate the cache by using operation-specific SQL queries, (iii) the actual comparison will later be done by code that does not compare XML documents but response objects. As explained in Section 5.2, this is an important aspect, which is very often implemented differently by related approaches. The comparison of objects makes the cache lookup more efficient and bypasses the undesired possibility that small changes in non-content-related details of irrelevant XML parts (e.g., headers) lead to retransmission of actually identical responses. The code 303 is returned when the proxy "thinks" that the response the client already possesses is still fresh. This feature is very useful when the calls to the original service need to be reduced, but it is out-of-scope for this thesis. It is only code 304 that can guarantee the freshness of the responses that the client has.

### 5.3.4 *Comparison with Related Work*

Based on the information provided in Section 5.2, as well as on further details that can be found in the referenced publications, Table 6 and Table 7 summarize the features of the approaches and imply the contribution of the approach presented in this thesis. As reference points, Table 6 and Table 7 also include the two simple, standard approaches, namely Direct Calls with Caching (DCC) and Direct Calls, No caching (DCN). As their names imply, they perform Web service calls with (client-side, TTL-based) caching and Web service calls without caching, respectively. It must be noted that the tables summarize *only* the features of the approaches that concern the

Table 6: Comparison of Web service response caching approaches with regard to their *freshness handling*

| CATEGORY | APPROACH | BASED ON | + | − |
|---|---|---|---|---|
| Provider–based | Liu & Deters, 2007 [79] | TTL, LFU, IRs from provider's cache | Three-fold mechanism | No freshness guarantee |
| | Li et al., 2008 [75] | Validity checks (comparison of response hashcodes) | Freshness guarantee | Hash function execution too costly for small responses |
| | Tatemura et al., 2005 [154] | No proposed freshness technique | No constraints for the client, all classic client-side mechanisms can be used | Freshness not considered |
| Mediator–based | Schreiber et al., 2010 [135] | Intelligent process-based prefetching | Prefetching is normally attempted shortly before actual calls | No freshness guarantee |
| | Elbashir & Deters, 2005 [34] | Service dependency matrix, "consistency-mode" (used only upon connectivity loss) | "Consistency-mode" guarantees freshness (but eliminates cache advantages) | No freshness guarantee in "performance-mode" |
| | *PCV, this thesis 2011 [104]* | *Validity checks (comparison of response objects)* | *Freshness guarantee* | *The client must understand the freshness-codes that replace the responses* |
| Client–based | Takase & Tatsubori, 2004 [152] | TTL | No constraints, TTL can be freely defined by users | No freshness guarantee |
| | DCC, standard approach | TTL | No constraints, TTL can be freely defined by users | No freshness guarantee |
| | DCN, standard approach | No freshness technique needed | Freshness guarantee | Freshness guarantee cancels all advantages of caching |

Table 7: Comparison of Web service response caching approaches with regard to their *ease of deployment for existing external Web services*

| CATEGORY | APPROACH | BASED ON | + | – |
|---|---|---|---|---|
| Provider–based | Liu & Deters, 2007 [79] | Cache manager modules and metadata for cacheability | New modules are transparent to the client app. | Architectural extension needed on both provider– and client–side |
| | Li et al., 2008 [75] | Hash function-enabled extensions | New modules are transparent to the client app. | Effort/extensions needed for the support of each new Web service |
| | Tatemura et al., 2005 [154] | New language for describing response generation, new provider-side middleware | The handling of the new language is abstracted by a middleware | Significant architectural extension needed on provider-side |
| Mediator–based | Schreiber et al., 2010 [135] | Mediating middleware | The mediating middleware is already at deployment-time compatible with any Web service | Support of many non WS-related technologies (Javascript etc.) assumed on client-side |
| | Elbashir & Deters, 2005 [34] | Method-call interception proxy | Possible both as embedded in client and as standalone proxy | Metadata about service/method dependencies needed from the provider |
| | *PCV, 2011 [104], this thesis* | *Automatically generated proxies based on the Web service description* | *No extensions (modules or metadata) needed on either provider- or client-side* | *New proxy must be (automatically) generated for each Web service* |
| Client–based | Takase & Tatsubori, 2004 [152] | Client-side Web services middleware | No extensions (modules or metadata) needed outside the client | Middleware must be installed on client |
| | DCC, standard approach | Client logic only | No extensions (modules or metadata) needed on either provider- or client-side | A client cache-logic must be implemented |
| | DCN, standard approach | No requirements | No extensions (modules or metadata) needed on either provider- or client-side | – (no constraints) |

Table 8: *Design decisions* of the current approach (PCV) that comprise key factors for its uniqueness compared to related work

| LEVEL | ASPECT | THIS APPROACH (PCV) | RELATED WORK |
|---|---|---|---|
| Concept & Goals | Nature of caching | Caching-support is provided as an adaptation mechanism for external, third-party Web services without access to their code or host system | Enhancements are suggested for Web services that can integrate caching as an inherent feature |
| | Focus of approach | Guarantee the legal use of always-fresh Web service responses without having to abandon caching | Tackle technical or architectural challenges to enhance the efficiency or the hit ratio of Web service response caching[a] |
| Cache functionality | Nature of the cache database | Automatically generated per proxy tailored to the objects used by each Web service | Pre-engineered and general-purpose for all Web services |
| | Compared cached entities | Application objects, in order to ensure that irrelevant XML header or metadata-representation details do not prevent the detection of cache hits | Web service responses as XML documents or similar[b] |
| | Caching state indicators | Injected in the form of service operation parameters[c] | Database keys, XML headers, or attached hashcodes |

*a* The approach of [74] comprises an exception, as it focuses on freshness and can provide it to almost 100%. However, its differences with regard to the rest of the aspects of this table, e.g., its pre-engineered cache, its (implicit) comparison of XML documents, and the fact that it does not focus on third-party Web services, make it inappropriate for a mediation layer.

*b* The approach of [152] comprises an exception, as it compares application objects, as well. However, automation and freshness –and, with that, all relevant design issues– are in that work completely out of scope.

*c* Referring back to the first difference between WWW and WS caching mentioned in Table 5, it is noted that this design decision is very important because it transfers the validity-checking process from a general-purpose Web server or database handler to the internal logic of the service operation itself. This fact remedies the main technical restrictions of the application of validity-checks in the Web service domain, as these restrictions are described in [159].

handling of freshness and the ease of deployment for existing, third-party services, because these are the features that help us identify and highlight the innovative aspect of PCV. Naturally, other approaches may have other advantages and focus on other aspects, but the latter are out of scope for this comparison.

The conclusion that can be drawn by a parallel study of Table 6 and Table 7 is that, in the domain of Web services, *PCV is the only approach that can enable the use of cached responses with a freshness guarantee generically for external, third-party services*. The use of cached or prefetched responses without establishing a connection to the server each time leads, by definition, to a risk of using outdated information. Following most of the described approaches, whenever 100% freshness is desired, caching must simply be deactivated. The authors of [135] explicitly state that caching or prefetching of critical responses must be avoided. This statement is true, unless the responses are verified before use and this verification concept lies in the heart of PCV.

Finally, Table 8 is a recap of important design decisions whose combination differentiates PCV from standard approaches. It is mainly due to the features explained in Table 8 why PCV has achieved its uniqueness, i.e., the uniqueness that has been stated in the previous paragraph.

## 5.4 EVALUATION

In the following, the PCV approach, which is enabled by the WSPG, is thoroughly evaluated regarding its effect on the performance of wireless Web service calls.

### 5.4.1 *Purpose and Setup*

PCV can be combined with other overhead reduction techniques and its usefulness is not restricted to the cases in which the devices act as SOAP clients. Indeed, throughout the practical use of the MML, PCV has often been combined with protocol-translation. The extra benefits are notable in these combined cases, as well. However, all other techniques are out-of-scope for this evaluation, whose purpose is to measure what PCV can contribute when used independently. Thus, all measurements refer to a scenario where devices, MML, and external services all communicate using SOAP, without any other optimizations.

The *evaluation setup* consists of the selection of the compared approaches, the definition of (dependent, independent, and controlled) variables, as well as the description of the experimental environment. After the description of the setup, some decisions concerning the controlled variables are explained in more detail separately.

- *Compared approaches:* PCV focuses on achieving 100% freshness and, therefore, it should be compared with the only existent approach for achieving this, namely DCN (cf. Section 5.2). However, PCV must also be compared with traditional client-side caching (i.e., DCC), in order to evaluate what has to be sacrificed for achieving this absolute freshness. Thus, three approaches are tested, namely DCN, DCC, and PCV. According to the expectations, DCC should performs best (but it risks using old data), while PCV may complicate the communication compared to DCN (PCV uses extra data for the validity check and replaces one service call with two) but it should reduce the wirelessly transmitted data in the case of a hit.

- *Dependent variables:* Two metrics are considered, which –depending on the scenario– may be most important to the client: The *reduction of the amount of wirelessly transmitted data* or *saved bandwidth* (sb) and the *user-perceived latency* (upl).

- *Independent variables:* As explained in Subsection 5.3.2, the main variables that determine the performance are the response size (s) of the Web service and the hit ratio (r). For PCV, the hit ratio is the probability of the cached response being still up-to-date, while for DCC it is, equivalently, the probability of using a cached response (of course, without knowing if it is up-to-date). s and r will be varied, while the rest of the variables will be controlled.

- *Controlled variables:* Other parameters that affect the results are the *client device*, the *network connection*, the *structure of Web service responses*, and the *workload* (or *service call pattern*). How these variables have been chosen/controlled/varied and why, is separately explained in the following.

- *Environment:* sb is exact and environment-independent. upl has been measured as response time plus parsing time, because it refers to the time between the user action that sends the request and the moment where the received results are ready to be presented to the user [135] (cf. also the similar definition of the metric referred to as "end-to-end delay" in [92]). Parsing times have been measured on a real device, while response times have simulated in order to ease tests with different networks (GPRS, UMTS, LTE). However, response times have often been validated through sample tests from the device.

*Setting the Controlled Variables*

The measurements were done with an iPhone 3G (iOS 4.0.2) *device*. Less capable devices would favor PCV even more compared to DCN, because even small bandwidth savings would lead to big differences in upl. Furthermore, even if more capable devices had been used, similar conclusions could be reached by slightly increasing the examined Web service response sizes, which would still be completely realistic. Thus, the mentioned device was exemplarily selected because it is popular and it obviously does not favor PCV.

With regard to the *network connection*, different results will be presented, namely for GPRS, UMTS, and LTE. EDGE and HSDPA perform similarly to GPRS and UMTS, respectively. WLAN can perform similarly to LTE, while the cases where WLAN achieves a performance almost equal to wired networks are obviously out-of-scope for any adaptation technique. Refer back to Subsection 2.2.2 for details about the mentioned access networks.

The experimental *Web services* have been chosen so that their response size can be easily varied. However, other service features may affect the results, as well. Responses with different structure may need different parsing times even for the same response sizes. Thus, two different real Web services were used: YellowMap's Point-Of-Interest

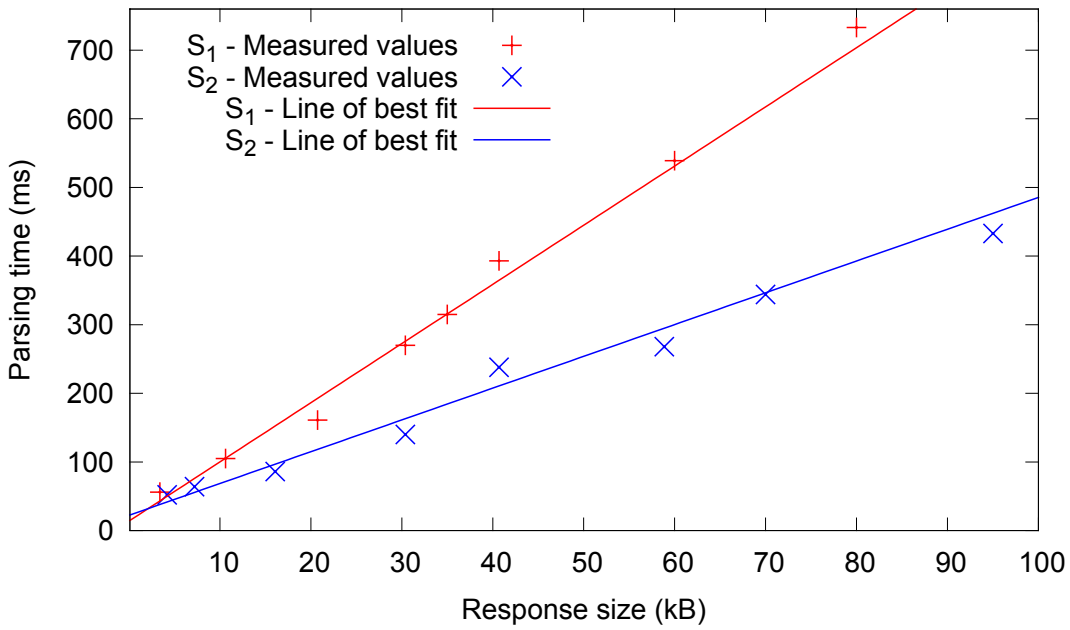



Figure 17: Examining the Web services with regard to their parsing complexity

service[3] ($S_1$) has high parsing complexity, while a SOAP version of Apple's "iTunes new releases" service[4] ($S_2$) is easy to parse.

Figure 17 shows the parsing times measured for different response sizes and the *lines of best fit* that represent the parsing time as a function of response size $s$. The latter functions were calculated with linear regression to be approx. $8.6228 \times s + 13.849$ and $4.632 \times s + 22.264$ for $S_1$ and $S_2$, respectively, given that time is measured in milliseconds and $s$ in kB. Other Web services would probably give a line somewhere between them, because, as mentioned, $S_1$ and $S_1$ are extreme cases in terms of parsing difficulty.

Different *workloads* have been tested. In the first part of the evaluation (evaluation of basic benefit, Subsection 5.4.2), most of the results are presented for a scenario where a number of clients just send the same request twice. The number of clients is irrelevant because the saved bandwidth and the user-perceived latency reduction will be presented relatively. The existence of many clients is only assumed (and simulated) in order to assign all possible values to the hit ratio. More precisely, the whole response is always fetched at the first request, while the second request leads either to retransmission of the response (cache miss) or to the transmission of the compact response. Thus, the savings cannot exceed 50%. This generic scenario was chosen because its triviality increases the comprehensibility of the results, its results constitute again a worst case (minimal repetition of identical requests) for PCV, and no other scenarios are commonly accepted as realistic [135]. This workload will be called *50–50*, indicating that 50% of the calls have a chance to –but will not necessarily– be satisfied by cached results.

3 `http://www.yellowmap.de/Partners/XML/PoiXmlServiceV21.asmx?wsdl` (Last accessed in January 2012).

4 `http://ax.phobos.apple.com.edgesuite.net/WebObjects/MZStore.woa/wpa/MRSS/newreleases/rss.xml` (Last accessed in January 2012).

In the second part of the evaluation (evaluation of the trade-offs under realistic Web service call traces, Subsection 5.4.3), the workloads refer to a single client that performs the same Web service call repeatedly but with different traces each time, i.e., with different assumptions about how often the service is called, how often its responses change etc. These workloads will be justified with references to related surveys, as well as with examples of mobile applications.

### 5.4.2 *Measuring the Basic Benefit*

The basic goal of PCV is to increase the saved bandwidth (sb). Both response time and parsing time reduction are actually caused by sb, so that upl depends up to some extent on sb. The question with regard to sb is for what values of s and r it becomes significant. Obviously, PCV cannot save much bandwidth when the exchanged messages are small and it is not worth it either when the expected hit ratio is low. For the 50-50 workload, Figure 18 represents sb in percentage of the originally used bandwidth, i.e., the bandwidth used in the case of DCN. The results prove that sb gets significant values early, e.g., ca. 25% of the bandwidth can be saved for mediocre values of r even when s is smaller than 10 kB (kilobytes). Note that sb is service-independent. The extra data used by PCV are normally trivial and may only cause minimal performance degradation for very small values of s and r (under the lowest delimitating line in Figure 18). In all other cases, the bandwidth savings are positive.

Figure 19 shows the reduction of upl with a similar representation, i.e., as a percentage of the values of upl without caching (DCN). As expected, the reduction of upl is bigger for $S_1$. Interesting is the fact that, depending on the relationship between parsing times and response times, the percentage of upl reduction may be similar for LTE and UMTS (compare $S_1$-UMTS with $S_1$-LTE in Figure 19). This is due to the fact that the parsing time sometimes dominates over response time in the overall upl, so that the connection becomes less important. The absolute upl reduction is, of course, bigger for UMTS, but the same does not necessarily hold when examining the relative reduction in percent. Most important is the fact that upl presents significant reductions even though the call is proxied and the backend (wired) call is, naturally, also included in the measurements. Figure 20 shows exemplarily and scenario-independently some absolute values for the upl reduction when $S_1$ is called with PCV (compared to DCN). A logarithmic scale is used because the reduction is much bigger for a GPRS network. Nevertheless, the upl reduction is significant for UMTS and LTE, as well, because enhancements of hundreds of milliseconds can be perceived by the user and thus cannot be considered to be trivial in this context.

The response sizes used throughout the evaluation were realistic and not very big. Both test services can give response sizes $> 500$ kB, while other Web services may cause even "heavier" communication. Frequent usage of such services causes much bigger amounts of data to be exchanged and it is easy to imagine how the basic benefit of PCV would become even more obvious in such cases.

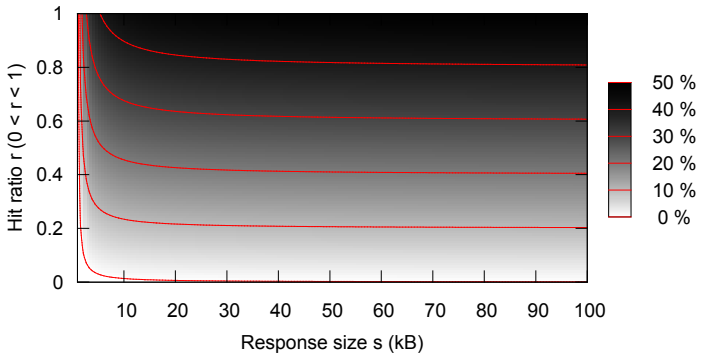
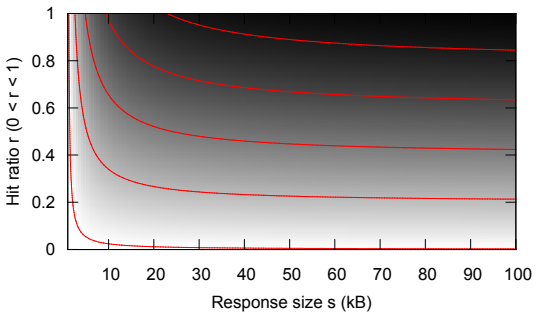
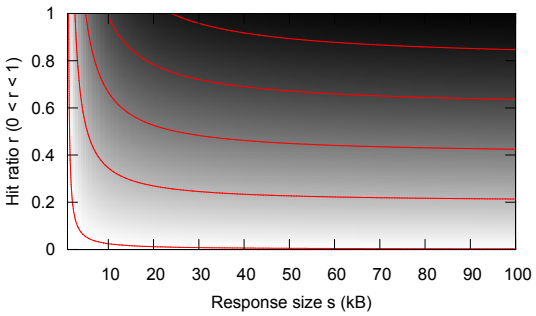


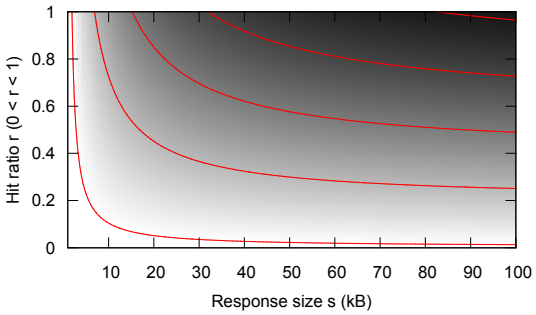

Figure 18: Relative saved bandwidth (sb) of PCV (compared to DCN for the 50–50 workload) as a function of response size (s) and hit ratio (r)



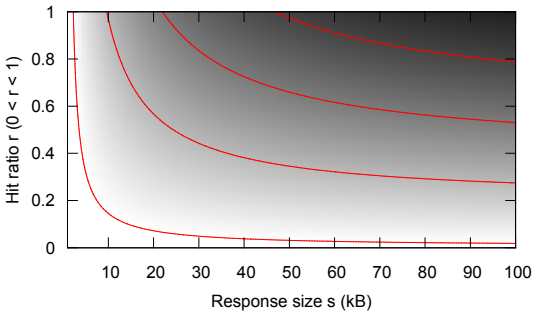

a) $S_1$ over GPRS



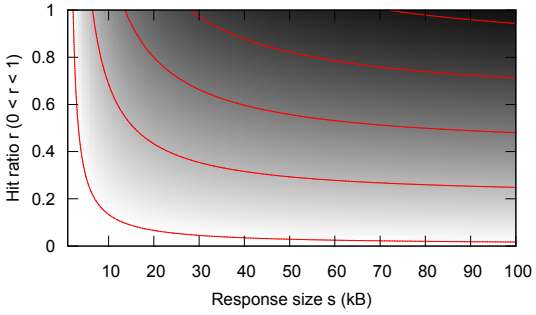

b) $S_2$ over GPRS



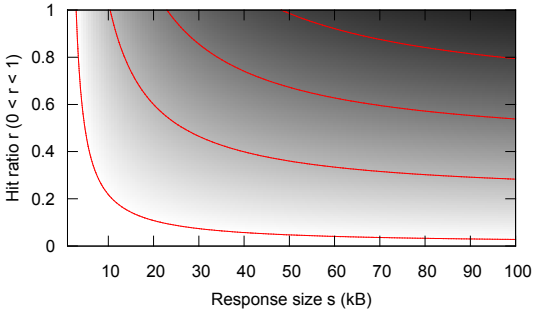

c) $S_1$ over UMTS




d) $S_2$ over UMTS




e) $S_1$ over LTE




f) $S_2$ over LTE

Figure 19: Relative user-perceived latency (upl) reduction of PCV (compared to DCN for the 50–50 workload) as a function of response size (s) and hit ratio (r)

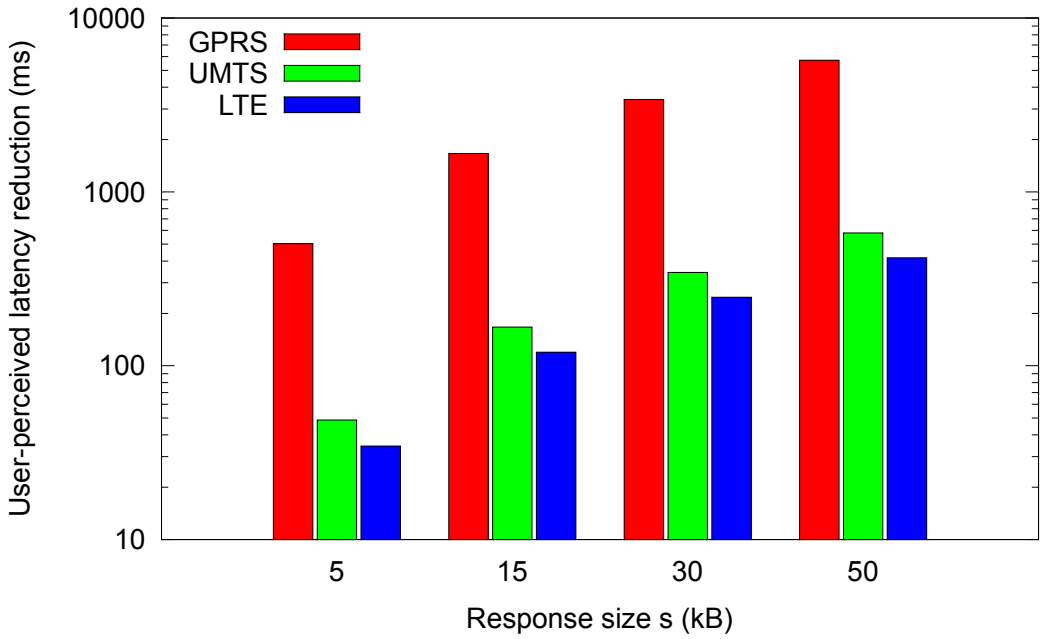



Figure 20: Examples of absolute user-perceived latency reduction (of PCV compared to DCN) for service S1

### 5.4.3 *Measurements with Realistic Web Service Call Traces*

The results of Subsection 5.4.2 have been limited to presenting the sb- and upl-reduction that PCV can achieve compared to DCN. DCC has not been involved yet because the fact that no service call actually happens at a cache hit renders response time equal to zero, so that it makes less sense to talk about upl. However, the following evaluation results will illustrate the spectrum between the performances of DCC and DCN where PCV is expected to lie, i.e., the trade-off *performance vs. freshness* between the approaches. In order to demonstrate the mentioned trade-off, an appropriate test dataset (workload) is needed. This datasets should normally consist of different Web service call traces, i.e., logs of the monitored activity of mobile applications that perform repeated service invocations. As also explained by [135], there are no such real public datasets because monitored activity of popular applications is usually private. Furthermore, finding a couple of such datasets would not be enough. They should also be proven to be representative. Taking this into consideration, the three approaches have been evaluated here with artificial traces, which are designed based on a use case analysis supported by related surveys. An infinite number of different traces could be designed and could appear in a real system. The traces used in the following are single incidents that are considered to be realistic and representative. The results must not be seen as a complete and exhaustive comparison.

The trace characteristics that are of interest –because they determine the performance of the approaches– are the *dynamicity* and the *response sizes*:

DYNAMICITY (d) is defined here as *the probability of each response to be identical with a previously sent response*. The dynamicity depends on two factors: The expected time intervals between subsequent service calls and the nature of the contents

of the response (i.e., if are they static or change often). Being a probability, the dynamicity is measured in percent (%).

RESPONSE SIZES (s) will have a different *spectrum* for each trace, depending on how big the responses of the trace are expected to be. Response sizes are measured in kB.

Use Cases (UC) of mobile, Web service-based applications that may perform repeated (identical) calls will be identified and discussed with respect to their expected dynamicity and response sizes:

A survey of Gartner, Inc. [115] identified the following top 10 consumer mobile application categories for the near future: i) money transfer, ii) location-based services, iii) mobile search, iv) mobile browsing, v) mobile health monitoring, vi) mobile payment, vii) near field communication services, viii) mobile advertising, ix) mobile instant messaging, and x) mobile music. Along with these results, it should be considered that the findings of two scientific surveys about public Web services [36, 75] can be summarized as follows: ca. 50% or more of the public Web services offer data lookup services, i.e., connections to –usually not very dynamic– databases, while the other important categories (10%-15%) are sensing services ([75] probably includes them in data lookup services), and data processing/conversion. After considering the categories of the Gartner survey, the mentioned types of public Web services, and some of the "hottest" application domains, the following UCs have been identified as interesting and relevant to the purposes of a caching evaluation:

- UC1: Data lookup, location-based service in the domain *mobility and transport*
  - *Example:* A mobile car-sharing application developed in the context of the project Green Mobility[5] includes a Web service request for Points-Of-Interest (POI). This request may be sent every time that the user wants to view possible meeting points on her/his map in order to choose one. Another good example from this domain would be a mobile monitoring application of a car rental company, which is informed periodically about the status (incl. location) of its cars through Web services.
  - *Characteristics:* The mentioned POI service sends responses that are usually some tens of kB but could also be much bigger. Similar could be true for a service that reports the status of cars, as location-based services in this domain may include similar format and information fields. Concerning dynamicity, responses may change because of newly added POI. This does not happen very often, but it should be considered that identical calls do not occur extremely often but between longer time intervals, a fact that obviously increases the dynamicity. In the car rental application, the time intervals may be much smaller (maybe some seconds) but the status of a car is more likely to change often, so that a given dynamicity is present there, as well. All in all, this UC is expected to have a medium dynamicity.
- UC2: Data lookup advertising service in the domain *ads in communication apps*
  - *Example:* One can think of a Web service request included in a messaging application. The request would periodically look for *current ads* (push

5 `http://www.greenmobility-project.de` (Last accessed in January 2012).

mechanisms may be used but are, especially for mobile apps, hindered by addressing problems or implementation difficulties, so that one could argue confidently that periodical pull is used very often). For example, the concept can be understood by considering Skype[6], where ads are continuously downloaded and shown at the bottom while the app is being used.

  - *Characteristics:* Ads may contain images, moving images, or even sound. The message sizes can vary a lot here, but big-sized messages must be considered here as a possibility, especially for the future. The dynamicity is not necessarily very high, because the request may be sent often, but the *current ad* will not change every few seconds. This would be bad for the advertised party and annoying for the user.

- UC3: Sensing service for mobile health-monitoring in the domain *assisted living*
  - *Example:* Many different example systems for this case can be found in [58]. According to the survey, the most widely monitored medical variables are the electrocardiograph, the heart rate monitor, the blood pressure monitor, and the oxygen saturation monitor.
  - *Characteristics:* Although sensor data are usually compact, a look at example electrocardiograph reports[7] reveals that a Web service response containing such a report could reach very big sizes (maybe hundreds of kB). However, if this degree of detail is needed, no chance for unchanged responses exists. Thus, only systems with summarized, smaller reports should be considered for caching. These systems do not report every detail, but the status. These have smaller sizes but a very low dynamicity, because the time intervals are very small (critical applications), while the summarized status has great chances of being identical for many subsequent invocations.

- UC4: Data processing/conversion service for *mobile payment systems*
  - *Example:* A classic currency conversion service[8] can be considered here, maybe combined with the processing of some extra user data.
  - *Characteristics:* Responses of such a service may normally not exceed a maximum of 2–3 kB. The dynamicity is also pretty high, because usual apps are not expected to perform many payments on a single day, while the response of the currency converter is normally modified at daily intervals.

In accordance with the discussion of the above UCs, Figure 21 coarsely depicts the characteristics that the Web service call traces of the UCs are expected to have. Based on these expectations, the following four traces have been designed in order to mirror the discussed UCs:

- Trace 1: Test run of Web service invocations with $d = 50\%$ and $s \in [20,80]$, where $s$ gets uniformly distributed random values.

6 An Internet Telephony Application, cf. `http://www.skype.com` (Last accessed in January 2012).
7 `http://en.wikipedia.org/wiki/Electrocardiography` (Last accessed in January 2012).
8 e.g., the public Web service at `http://www.webservicex.net/CurrencyConvertor.asmx?WSDL` (Last accessed in January 2012).

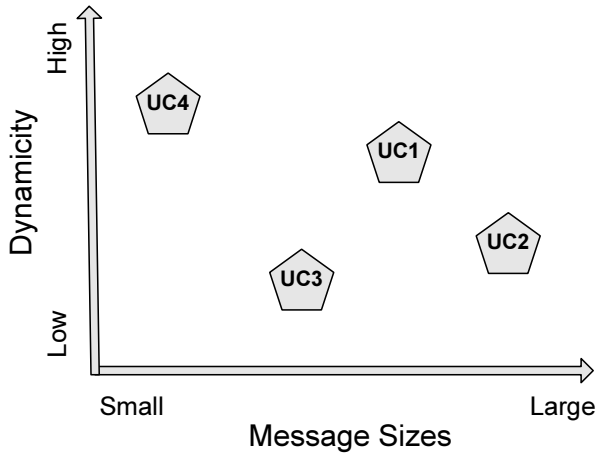


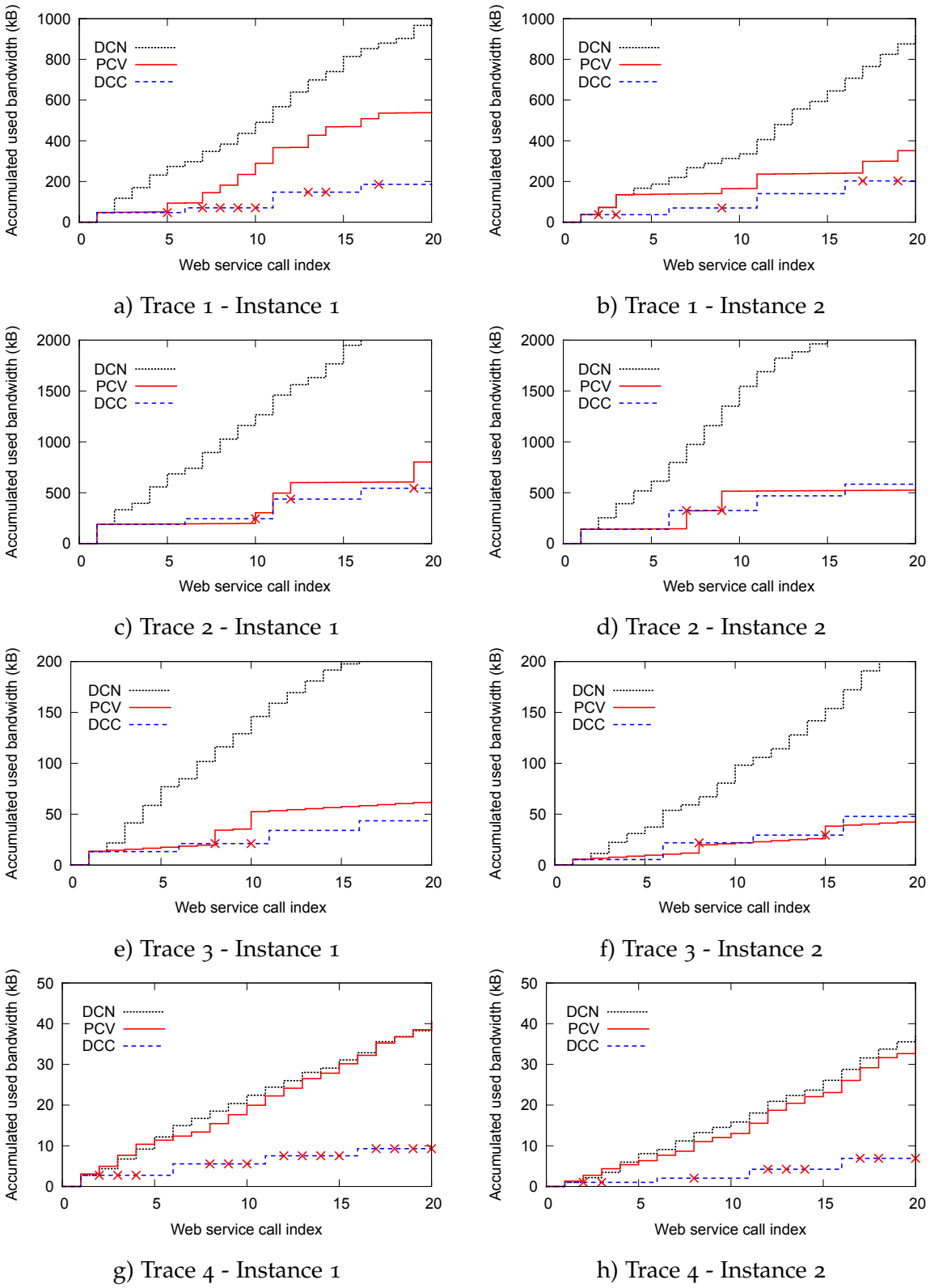

Figure 21: Expected dynamicity and message sizes of the Use Cases of interest

- Trace 2: Test run of Web service invocations with d = 20% and $s \in [50,200]$, where s gets uniformly distributed random values.
- Trace 3: Test run of Web service invocations with d = 10% and $s \in [5,20]$, where s gets uniformly distributed random values.
- Trace 4: Test run of Web service invocations with d = 70% and $s \in [1,3]$, where s gets uniformly distributed random values.

Figure 22 shows for each trace the accumulated bandwidth used by DCN, DCC and PCV for 20 subsequent identical requests. DCC uses for its cached objects a TTL such that the objects expire after a time that corresponds with 5 calls, so that the 6th call has to fetch a new response. TTL is normally used as part of DCC [20], while the particular value has been chosen so as to be appropriate enough for the examined cases. Other values, which make DCC look less efficient, have been tested but are not included, because they do not offer many interesting new insights. Even so, it is easy to imagine how altering the TTL would alter the results. Each trace has been run many times, thus providing different instances. Averaging the results would be wrong and meaningless, as they already contain many invocations with random values. Instead, we select two typical instances of each trace and discuss their meaning.

After a first look upon all the results, it becomes obvious that for $TTL < \infty$, PCV can even achieve better results, i.e., use less bandwidth, than DCC. However, this happens only in cases in which the responses change rarely and, in such cases, DCC would not use many outdated information. Even in the normal case, where PCV uses more bandwidth than DCC, PCV often achieves a bandwidth usage close to that of DCC. Furthermore, the crosses denote invocations for which DCC would have used outdated data. It is noted again that the only other approach that can prevent these red crosses from occurring is DCN, whose results are depicted, as well.

More concretely, Figure 22a and Figure 22b show that, for Trace 1, the bandwidth consumption of PCV remains somewhere between those of DCN and DCC. Indeed, this was true for all instances of Trace 1. However, the appearances of (red) crosses in the case of DCC are so frequent that PCV should definitely be considered, at least for critical applications.




a) Trace 1 - Instance 1

b) Trace 1 - Instance 2

c) Trace 2 - Instance 1

d) Trace 2 - Instance 2

e) Trace 3 - Instance 1

f) Trace 3 - Instance 2

g) Trace 4 - Instance 1

h) Trace 4 - Instance 2

Figure 22: Accumulated bandwidth used by the three caching approaches (PCV, DCC, DCN) for the different traces. Red crosses indicate the use of outdated responses

Traces 2 and 3 are cases where PCV has very high chances of being the preferred approach. Contrary to Trace 1, the difference of the used bandwidth between PCV and DCC is usually insignificant and there have even been cases where PCV consumed less bandwidth in total, although DCC has used a couple of outdated responses (Figure 22d and Figure 22f). However, the used bandwidth is not the only metric that may be of interest. If, for example, the *number of established connections*, which may affect response times and/or energy consumption, is more important than the freshness of the responses, then DCC may remain preferable.

As shown by Figure 22g and Figure 22h, Trace 4 reflects cases where PCV is not efficient, at least in terms of bandwidth usage. In the corresponding instances, PCV has used almost as much bandwidth as DCN. Sometimes, PCV uses even more bandwidth than DCN (cf. Figure 22g), because in the case of small messages (combined with low hit ratio), the overhead of the extra fields used by the proxy is not insignificant compared to the size of the content field of the response. Analogously, the data whose re-transmission is spared by PCV, is sometimes no more than half the size of the response, which is transmitted anyway.

Concerning the impact that these differences of used bandwidth may have on the application in terms of user-perceived latency, the reader may refer to Subsection 5.4.2.

### 5.4.4 *Discussion*

With retrospect to the research question formulated in Section 5.1, taking all evaluation results into account, the following statements can be inferred:

- Concerning the *applicability and generality of the mechanism*: The fact that the only input needed for the application of the mechanism are the (standardized) service descriptions indicates that the solution is generically applicable to external Web services. Furthermore, the evaluation has been performed with existing third-party services, while proxies have been successfully generated for other Web services, as well. Although it cannot be proven that the implemented proxy generation would work for *every* supported Web service, the approach is generic because is not based on any service-specific features or implementation details and it does not assume access to external systems.

- Concerning the *performance enhancement of PCV compared to other approaches that guarantee 100% freshness*: Although the universal statement "the proposed approach performs always better than other approaches that guarantee 100% freshness" cannot be made (cf. Figure 19, Figure 22g, Figure 22h), the comparisons of PCV with DCN definitely proves that PCV performs (in terms of saved bandwidth and user-perceived latency) better than DCN for a very wide spectrum of values of hit ratio and response sizes (cf. Figure 19) and for a variety of realistic Web service call traces (cf. Figure 22).

- Concerning the *qualitative advantages of PCV compared to approaches that risk freshness in order to enhance performance*: The results of Figure 22 show that, in realistic scenarios, PCV not only avoids many occurrences of outdated Web service response usage, but also that its performance in terms of used bandwidth is very close to the performance of standard freshness-risking approaches, while it may sometimes perform even better than them.

Having completed the discussion of the most important findings of the conducted evaluation, some final remarks of this discussion should be dedicated to the goals that have *not* been pursued and the metrics that have *not* been investigated. This refers to metrics that can normally be enhanced with the use of caching, but its detailed examination has been out of scope due to the subject of this work. As this work has focused on *performance* of Web service usage on *mobile and wireless* devices, *consumed bandwidth* and *user-perceived latency* have been of interest, while *energy consumption*, *monetary costs*, and *server load* have been only marginally relevant:

- *Energy consumption* is very implicitly perceived by the user. The user only notices how often the battery of a mobile device has to be charged, which is not counted in milliseconds or seconds, but in hours, days, or even weeks. Thus, energy consumption is normally not considered to be a performance indicator. However, it is relevant for most mobile and wireless devices and, theoretically, it can indeed be influenced by the use of caching. For this reason, in the margin of the presented evaluations, some energy-related measurements have been performed. The corresponding results lead to generally interesting findings and indicate open issues for research. However, they are only attached and described in Section A.2 because of three reasons: Firstly, the variations in energy consumption caused by the use of caching have been found to be much less significant than those caused by other factors, such as the time intervals between the Web service invocations, or the characteristics of the devices and the networks used. Secondly, the caching approaches examined in this chapter were not directly compared to each other because saving energy has not been the primary goal of their design. Thirdly, as already mentioned, performance is the main focus of this work and energy consumption is not a performance indicator. However, the results have indicated that PCV may be close to DCC also in terms of energy-efficiency, as the elimination of connection establishments between the device and the Web service provider, which is achieved by DCC, did not seem to contribute much in saving energy under the given experimental setting (cf. Section A.2).

- *Monetary costs* are completely irrelevant to performance and depend much more on other factors, e.g., service pricing models [32]. Even in cases of pay-per-use service invocations, the use of cached responses (in order to save money) is very often explicitly prohibited. In such a case, PCV would be the only option for the exploitation of cached responses, because it does not violate this prohibition, as it invokes the original service every time when a response is needed.

- *Server load* is at first sight irrelevant to mobile and wireless devices that act as Web service consumers. Of course, if a provider gets overloaded, the response times experienced by the mobile users may, in turn, become longer. However, the handling of this challenge lies rather on the side of the provider. Further, in the face of Cloud Computing, huge data centers, virtualization, and load balancing techniques, the way some (probably isolated) mobile clients use caching (or not) is unlikely to have a significant effect on the total server load.

## 5.5 SUMMARY AND CONCLUSIONS

This chapter has presented a Web service adaptation mechanism that is based on the idea of enabling response validity-checks for external Web services, so that devices can use their old responses when they are still fresh, thus avoiding the wireless transmission of complete responses during each Web service invocation.

The exploration of related work in Section 5.2 has revealed why this new adaptation mechanism can offer a combination of advantages that cannot be provided by any existing mechanism. These advantages relate to the guarantee of 100% freshness for used cached responses, combined with the applicability of the mechanism for any external Web service without access to its hosting system or source code.

The most important details with regard to the development of the proposed solution are the following: The enabled Web service communication scheme, the workflow for the generic and automated generation of proxies that enable the mentioned communication scheme, and the algorithms that are used in order to fulfill critical tasks of this workflow. These details have been explored and explained in Section 5.3.

The described solution has been evaluated through comparison with the standard approaches for (i) achieving 100% freshness and (ii) enhancing performance by risking freshness. Compared to the first ones, the proposed solution presents significant performance enhancements for a wide spectrum of system conditions and under a variety of realistic Web service call traces. Compared to the second ones, it remains very close to them in terms of performance –sometimes even enhancing it– while it does not risk the freshness of the responses as they do.

# 6

# DECISION SUPPORT FOR WEB SERVICE ADAPTATION

»The percentage of mistakes in quick decisions is no greater than in long-drawn-out vacillation, and the effect of decisiveness itself makes things go and creates confidence.«

— Anne O'Hare McCormick

Both of the two previous chapters form part of the proof that many different Web service adaptation mechanisms are possible to implement and almost all of them have reasons to exist, i.e., may be the preferable mechanisms under particular circumstances. For a mediation layer that enables, offers, and/or enforces such adaptation mechanisms, this truth already implies the existence of a decision problem. The current chapter analyzes this decision problem by formulating it mathematically, by suggesting algorithmic solutions that are based on insights of Chapter 4, and by examining the possibilities for making these solutions resistant to data missingness.

## 6.1 RESEARCH QUESTION AND SCOPE

In order to specify the examined issue and narrow the decision problem, it is noted again that, in the reference scenario of this thesis, the Web service adaptation mechanisms are enabled by proxies, which are generated on a mediation layer for existing external Web services (cf. Section 4.2 and Section 4.3), while the suitability of the mechanisms depends on aspects of the system context (cf. Section 4.4). Thus, the decision question of "*Which adaptation mechanism should be applied for each Web service under particular conditions?*" is translated to the decision question "*Which proxy should be generated/activated for each Web service under the current system context?*". But what are the research questions that arise when corresponding decision support has to be added to the mediation layer? The following paragraphs explain the main idea and the research question that drive the research presented in this chapter, as well as some assumptions and limitations about the environment for which this research is meaningful.

### *Main Idea and Research Question*

The main idea of this chapter is to design a decision support approach which is based on the results and insights of the survey of adaptation mechanisms that has been presented in (cf. Section 4.4). It is assumed that decisions based on these insights are educated and thus reasonable. Therefore, the focus of the evaluation is not laid on examining how good the decisions are. This is also because further restrictions make the direct comparison of decision support algorithms difficult or even meaningless[1]. Instead, the evaluation focuses on enhancing the proposed algorithms by tackling

1 The restrictions and further reasons that determined the focus of the comparisons and evaluations are explained in more detail in Subsection 6.5.1.

the problem of missing data scenario-specifically. The latter problem has proven to be very relevant for a mediation layer such as the MML and is one of the major research challenges of modern self-adaptive systems[2]. Thus, the research question is formulated as follows:

*Given an external Web service and a set of possible proxies, how can the suitability of each proxy for this Web service be scored based on the system context and what is the best way to make this scoring resistant to missing context data?*

The examination of this research question evidently includes performing the following tasks:

- Concretization of the problem of deciding between different Web service adaptation mechanisms, i.e., proxies.
- Development of solutions for solving the concrete decision problem.
- Evaluation of the possible ways for making the designed solutions resistant to missing context data.

These tasks are handled in the remaining parts of this chapter as follows: Section 6.2 handles the first task by providing a mathematical formulation of the examined decision problem. Section 6.4 handles the second task by describing two corresponding proxy scoring algorithms for decision support, one QoS-based and one QoE-based. Section 6.5 handles the third task by performing a thorough statistical evaluation of the most important imputation algorithms that can be used in order to maintain the quality of the scoring results in scenarios with missing context data. Section 6.3 explores related work and open research issues that are relevant for both the development of scoring algorithms (Section 6.4) and the evaluation of imputation algorithms (Section 6.5).

*Assumptions and Limitations*

Concerning general system features but also specific details of the proxy-based adaptation scenario, the following assumptions should hold about the technical environment for the decision support algorithms of this chapter to be meaningful:

- *Correctness of the underlying survey:* The fact that the proposed algorithms are tailored to the characteristics of the results of the related survey (i.e., the survey presented in Section 4.4) supports the argument that they provide an educated scoring. This is also one of the reasons why they are not directly compared to alternative decision support algorithms or to each other. Therefore, the correctness of the methodology used throughout the survey, but also of the results of the survey as such, is a requirement for the decision support algorithms to be considered useful.
- *Restrictions for the operation of too many proxies:* A basic assumption is that the mediation layer that hosts the proxies, e.g., the MML, cannot afford (or is not

2 Dealing with incomplete information and uncertainty is mentioned as an important research challenge of self-adaptive systems in both [129] and [25], which are the most recent research roadmaps for self-adaptive systems.

willing) to generate and operate every possible proxy for every known external Web service. In order to understand the validity of this assumption, one should consider that space and security concerns arise from the operation of many proxies. Concerning space, it is obvious that, for each Web service proxy, new modules have to be deployed somewhere, e.g., on a Web container, on an RMI registry, or similar. Concerning security, it is noted that the offering of each new proxy means the opening of new interfaces and ports, which need maintenance and administration and are often vulnerable to attacks. Furthermore, an infinite number of Web services may be found on the IoS. Thus, a mediation layer operator normally desires to avoid the generation and operation of unnecessary proxies. Practical experience from the operation of the MML has also pointed to this fact. However, the possibility of a mediation layer that is not concerned at all with the mentioned issues cannot be excluded. In the latter case, the decision support would be useless, although some of the research results could be interesting for a similar scenario, where particular clients would, for example, need to decide which of the available proxies to use. However, the current work researches the issue from the mediator's perspective.

- *Client-proxy compatibility:* In order to be general, the related survey has avoided including device details, which may change from year to year or from a release of a mobile operating system to the next one, and has focused on always-important system contexts. However, it may occur that particular clients are incompatible to certain proxies. This fact is not completely ignored (e.g., it is implicitly considered by the QoE-based algorithm), but, generally, it is assumed that the client devices are able (i.e., technically equipped) to use the generated proxies.

## 6.2 FORMULATION OF THE DECISION PROBLEM

The decision problem handled in this chapter is referred to as Always Best Served (ABS), in accordance with the well-known and much investigated Always Best Connected (ABC) problem [46, 175]. More information about their similarities and their differences is provided in Section 6.3. While ABC scores available access networks, ABS scores possible proxies. The outcome, i.e., the scoring, of the alternative proxies can be used for diverse purposes. For instance, the scores may be seen as suggestions to system operators or they may be used for the automated triggering of proxy generations. These diverse ways of exploiting the scores are out of scope and will not be further discussed, for the sake of simplicity. Instead, the scoring algorithms are assumed to be part of a general-purpose decision support system. The latter is provided with information about the proxy characteristics and about the past Web service usage, and is expected to suggest how suitable each proxy would be for a service.

The formulation of a problem that is to be solved by a scoring algorithm consists of exact descriptions of the input and the expected output. Related descriptions and definitions are provided in the following.

Table 9: Possible proxies and their characteristics

| Proxy | Bandwidth ($a_1$) | Latency ($a_2$) | Packet loss ($a_3$) | Stability ($a_4$) | CPU power ($a_5$) | Data size / SOAP size ($a_6$) | SOAP message size ($a_7$) | Processing time ($a_8$) | Service call frequency ($a_9$) | Service call criticality ($a_{10}$) |
|---|---|---|---|---|---|---|---|---|---|---|
| $p_1$ | u | $\geqslant$m | $\leqslant$s | u | u | u | $\leqslant$m | $\leqslant$s | u | $\leqslant$s |
| $p_2$ | $\leqslant$m | u | u | u | u | $\leqslant$m | $\geqslant$m | u | u | u |
| $\vdots$ | $\vdots$ | $\vdots$ | $\vdots$ | $\vdots$ | $\vdots$ | $\vdots$ | $\vdots$ | $\vdots$ | $\vdots$ | $\vdots$ |
| $p_N$ | $\in T$ | $\in T$ | $\in T$ | $\in T$ | $\in T$ | $\in T$ | $\in T$ | $\in T$ | $\in T$ | $\in T$ |

*Input*

With respect to the nature of the data that shall determine the proxy scores, and taking the involved entities and attributes, their possible values, and the missing data issue into account, the following sets and variables are defined:

- $P$, as the set of possible proxies, with the elements $p_i \in P, i = [1,N], i \in \mathbb{N}$, where $N$ is the number of possible proxies.
- $R$, as the set of Web service call records (i.e., monitored Web service invocations), with the elements $r_i \in R, i = [1,K], i \in \mathbb{N}$, where $K$ is the number of records.
- $A$, as the set of attributes used for characterizing the elements of $P$ and $R$, with the elements $a_i \in A, i = [1,11], i \in \mathbb{N}$. $A$ has 11 elements, corresponding with the context attributes derived from the analysis of Chapter 4.
- $V$, as a 4-set that consists of the possible values of the elements of $A$, namely $V := \{s, m, h, u\}$. Refer to Section 4.4 for the meaning and order of $s$, $m$, and $h$, while $u$ denotes an unknown value. In order to denote thresholds based on these values, a set of thresholds is also defined as $T := \{\leqslant, \geqslant, \emptyset\} \times V$.
- $S$, as the set of known external Web services, with the elements $s_i \in S, i = [1,L], i \in \mathbb{N}$, where $L$ is the number of Web services. Although $R$ has been previously defined service independently (and may exist as such), each service $s_i$ may have its own set of records, which is then denoted as $R_i$.

Table 9 and Table 10 visualize the defined sets and variables, providing example instances, as well as indicating the value ranges. Note that while the attribute values of the elements of $R$ (cf. Table 10) are always elements of $V$, the values of the corresponding proxy features (cf. Table 9) are expressed with the help of the same values, but the values are accompanied by the symbols $\leqslant$ and $\geqslant$ (cf. also Table 4 in Section 4.4). The extra attribute $a_{11}$ used in Table 10 indicates which proxy has been

Table 10: Monitored Web service call records and their characteristics

| Record | Bandwidth ($a_1$) | Latency ($a_2$) | Packet loss ($a_3$) | Stability ($a_4$) | CPU power ($a_5$) | Data size / SOAP size ($a_6$) | SOAP message size ($a_7$) | Processing time ($a_8$) | Service call frequency ($a_9$) | Service call criticality ($a_{10}$) | Chosen proxy ($a_{11}$) |
|---|---|---|---|---|---|---|---|---|---|---|---|
| $r_1$ | u | m | s | u | u | u | s | s | s | m | $p_4$ |
| $r_2$ | u | u | h | u | h | u | m | s | m | h | $p_1$ |
| ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ |
| $r_K$ | $\in V$ | $\in V$ | $\in V$ | $\in V$ | $\in V$ | $\in V$ | $\in V$ | $\in V$ | $\in V$ | $\in V$ | $\in P$ |

Table 11: Example output of a scoring algorithm

| | $p_1$ | $p_2$ | ... | $p_N$ |
|---|---|---|---|---|
| $s_1$ | 0.1 | 0.67 | ... | -0.55 |
| $s_2$ | 0.33 | -0.5 | ... | 0.8 |
| ... | ... | ... | ... | ... |
| $s_L$ | $\in B$ | $\in B$ | $\in B$ | $\in B$ |

selected for the recorded service invocation and shall only be relevant when user choices are considered (QoE-based approaches). The information contained in these two tables is the input to be given to a scoring algorithm.

*Output*

What is needed as output is a set of scores, each score corresponding to a service-proxy pair ($s_i$,$p_j$). The range and the meaning of the scores themselves depend on the algorithm that calculates them. Thus, the range of scores is abstractly defined here as $B := [b_{min},b_{max}]$, where $b_{min} \in \mathbb{R}$ is the minimum score and $b_{max} \in \mathbb{R}$ is the maximum score. Normally, the bigger the score, the more suitable the proxy for the respective service. Table 11 visualizes an example scoring output for $b_{min} = -1$ and $b_{max} = 1$. As already explained, the exact way in which the mediation layer (or its operator) uses these scores, is open and out of scope for this thesis.

## 6.3 RELATED WORK

The described decision problem is called ABS, because a theoretical optimal solution would help achieve the goal of always serving the wireless clients with the best possible (adapted) way to consume a Web service. *The focus of the research is on*

*designing decision support algorithms that match the peculiarities of the problem and on making them as resistant as possible against the existence of missing context data*. Therefore, after discussing the relationship between ABS and ABC, decision support algorithms from related domains and the state of the art for handling missing data in engineering are described in order to identify open issues, for which the remaining parts of this chapter will offer insights.

*ABS and ABC*

ABC is a well-known and much investigated issue, concerned with letting wireless devices switch among different access networks that they can possibly use (e.g., WLAN, UMTS, GPRS, or Bluetooth, cf. also Section 2.2). The goal is, of course, to choose each time the access network which is most appropriate in the current context. Research on the ABC problem over recent years presents an evolution which is driven by the constant appearance of new alternatives for handling connectivity in the wireless world, always accompanied by new challenges. After briefly portraying this evolution, it is explained how ABS appears with a similar logic but on another level, presenting similarities, but also differences, compared to ABC.

The term ABC has been probably introduced in [46]. This publication focused mostly on the scenario, the architecture, and the involved actors, while it referred to technologies of second- and third-generation mobile networks. Decision support (i.e., "Access Selection") is mentioned, but not really examined as a well-defined decision problem. The latter has been performed by a variety of works that appeared on this issue in the years that followed, e.g., [37] and [174]. With the recognition of a series of limitations of the involved network access technologies, the decision problem takes more concrete forms, modeled for example as a knapsack problem with characteristics and constraints derived from the technology-related limitations [41]. The evolution of the issue does not stop there, so that [175] re-formulates and re-examines the problem in order to address the peculiarities of fourth-generation mobile network architectures. For example, [175] categorizes the fourth-generation network attributes involved in the decision process in those that should be maximized, those that should be minimized, and those that should exceed certain thresholds. It also considers personalization of the decision process through user weighting, as devices and network usage start to vary heavily among modern users. The described evolution of the ABC problem indicates that new technological or architectural features generate a demand for new approaches for providing corresponding decision support. Indeed, new algorithms continue to appear for the ABC problem, re-considering, for example, QoE [27], user preferences and new testing environments [141], and more.

ABS, as previously described, is a similar problem which appears when moving up in the Open Systems Interconnection (OSI) model [181], from the network layer to the transport and session layers. There, instead of access networks, access methods to Web services have to be selected. However, the handling of ABS presents even more *differences* to ABC than the various alternative formulations of ABC to each other, namely:

- Appearing on a different layer, ABS needs partially different context information (cf. Section 4.4). Depending, firstly, on the granularity with which the context information can be rated and, secondly, on how deterministically the context

pinpoints the appropriate alternatives, certain approaches may be adequate for solving the one problem, but not the other.

- The conditions that make a proxy adequate for use have been researched to much less extent than the conditions that render access networks adequate. This is indicated by the big number of works that have already appeared for the handling of the ABC problem (cf. previous paragraph).
- Because of the time required for the proxy generation and because of the complex logic that would be otherwise needed on the client-side, the decisions in ABS are normally not taken per device and on-the-fly (or "real-time") for particular tasks, but they rather refer to (and affect) a set of devices. As a result, it is less meaningful to talk about optimization in the case of ABS. The reason is that in ABC, a device may have all the information that it needs in order to optimize the selection of an access network for a given action. In ABS, a proxy is generated for future usage and for many devices, the exact characteristics of which are unknown.

*Decision Support Algorithms in Related Domains*

ABC is, of course, not the only domain related to networks or computing in which similar decision support or scoring algorithms have been developed. For example, [148] provides a detailed analysis of QoS- and QoE-Management for UMTS cellular networks, where decision algorithms play an important role. [148] is not concerned though with ABS-specific attributes such as the characteristics of Web services and it rather presents solutions for problems such as routing or mobile network configuration. Concerning, for example, the structure of the used context and the types of knowledge incompleteness that are likely to appear, ABS obviously presents many differences compared to the problems discussed in [148]. Further interesting scoring algorithms can be found in the domain of event-detection. For example, [182] scores the relevance of events detected by sensors in order to decide if they should be propagated to decision makers or not. However, the scoring of [182] is based on structured score sheets and decision-maker weightings of the goals. Not only would such score sheets be impractical in the ABS scenario, but the goals (used bandwidth, user-perceived latency, energy consumption) are also so close to each other that putting weights on them would not necessarily change the results dramatically, not to mention that no detailed analysis such as this of Section 4.4 is available for the specialized goal of reducing energy consumption.

All in all, decision support or scoring algorithms may be used in any domain and may be based on any mathematical foundations. As the focus of this chapter is *not* on providing optimal decisions of any kind, but rather on designing algorithms that match the qualitative characteristics of the problem and on examining and enhancing their behavior against missing data, the examination and comparison of any further scoring approaches is here out of scope.

*Handling of Missing Data in Engineering*

As explained in Subsection 2.3.3, missing data may appear in data sets such as those generated by system monitoring modules. The approaches used for handling such missing data are called imputation algorithms (or imputation strategies). An approach that attempts to tackle the problem at its roots is to use scenario-specific correlations of the missing data in order to repair the sources of the errors. For example, [171] presents an approach for identifying erroneous sensor data sources by analyzing the missing data in combination with knowledge about the business logic of the system. Such solutions are, however, not considered here, because it is assumed that data missingness cannot be avoided and that the control over the error sources (user devices, mobile networks) is low or completely absent.

Thus, the focus here is laid on imputation algorithms. The state-of-the-art imputation algorithms are general-purpose and their success when they are applied in certain domains depends on the peculiarities of the domain. Therefore, scientists of different domains re-examine and/or try to enhance the suitability and the performance of particular algorithms in their domain. In the following, a list of examples where imputation has been revisited for particular cases in the engineering domain is provided, while the subsection concludes with remarks that explain which special features of ABS are likely to determine the performance of different imputation algorithms:

- [173] points out that in a database consisting only of attributes of type "char" (character), which have nominal categorical values, simple association rules for the character values may perform better imputation than the state-of-the-art (statistical) "nearest-neighbor" approach. For details about the nearest-neighbor approach, the reader can refer to [87].
- [47] points out how the fact that sensor data are spatio-temporally related affects the estimation accuracy of different imputation techniques, but also that complex imputation techniques (e.g., multiple imputation, cf. Subsection 2.3.3) would not be feasible in an environment where only few (the most recent) of the past records can be used (because of memory constraints or similar).
- [76] points out that for sensor data which present spatio-temporal relationships, a technique that is aware of the related spatio-temporal patterns can outperform, for example, state-of-the-art algorithms based on the maximum likelihood concept (cf. Subsection 2.3.3).
- [128] points out that for (audio) data collected for speaker identification, where missing data are common, certain records can be marked as unreliable, so that imputation techniques that ignore them should perform better.
- [180], [10], and [55] also revisit the issue of data imputation in the domains of vehicle traffic, user movement, and sensor databases, respectively. However, they are less concerned with standard imputation algorithms, but they rather focus on scenario-driven heuristics. Therefore, they are not individually discussed.

All the referenced cases presented peculiarities that determine what kind of imputation is more efficient for their data. A closer look at the ABS issue reveals the following aspects that should play a role in determining the efficiency of different imputation algorithms in the context of ABS:

- ABS involves the use of categorical data (as in [173]) and, more specifically, discrete categorical data with a very restricted value range. This fact may affect the importance of particular statistical properties (e.g., mean values, standard deviation) used by the imputation algorithms.
- ABS presents its own types of missingness, which are not expected to be MCAR (cf. Subsection 2.3.3). The monitored records are rather expected to have dependencies which may be explicitly or implicitly caused by scenario-specific properties such as: "More capable devices are expected to use better connections", "Records from better connections are expected to include less network-related missing data", etc. The possibilities that the monitored data have such characteristics will be analyzed in more detail and reflected in the evaluation test cases.
- In ABS, scenario-related decision support algorithms should run upon the imputed data. The nature of the algorithms that run upon the handled data is also critical for the efficiency of an imputation algorithm. It must be noted that imputation algorithms may be evaluated either by measuring how correctly they recover missing values or by measuring how well they eliminate the effect of missing data on an algorithm that uses the data after the imputation. It is *not* necessarily true that imputation algorithms which perform better with respect to the first criterion (i.e., perform a more exact guessing), also minimize the error of the *final* result (i.e., after some algorithm runs upon the imputed data). [87] states accordingly that "it is probably a popular misunderstanding that the goal of imputation is to predict individual missing values".

*Open Issues*

Driven by the analysis of the related work presented in the previous paragraphs, the open issues identified in relation to ABS can be summarized in the two following points:

- Development of decision algorithms, or adjustment of existing ones, with the following two properties:
  - They can solve the problem exactly as it is formulated in Section 6.2, delivering meaningful results, while also representing the state-of-the-art general approaches for offering decision support, namely QoS and QoE (cf. Section 2.3).
  - Their logic is intuitively derived by the characteristics of ABS in terms of used context attributes, involved value ranges, etc. The latter are determined by the granularity of the results of the related survey (cf. Section 4.4).
- Thorough evaluation of the application of different imputation algorithms in ABS, in order to examine, firstly, which state-of-the-art solutions should be preferred and, secondly, if the investigation of new imputation techniques tailored to the ABS scenario is worthwhile.

## 6.4 PROPOSED SCORING ALGORITHMS FOR DECISION SUPPORT

In accordance with the definitions of QoS and QoE, two principally different categories of scoring algorithms can be developed for decision support. A QoS-based scoring algorithm for decision support would rate each proxy by comparing its characteristics (cf. Table 9) with the monitored service call records in order to see how well the proxy would match the respective invocations. On the other hand, a QoE-based approach would focus on past user decisions, i.e., it would perform the scoring by analyzing the relationships between the values of the parameter $a_{11}$ ("chosen proxy") and the values of the other attributes. The correspondingly developed scoring algorithms are described in the following. *For the used notation, refer back to Section 6.2*, as well.

### 6.4.1 *Quality of Service-based Scoring Algorithm*

As is usually the case in QoS-related theory, the proposed QoS-based algorithm uses a utility function for the scoring of the different options. This utility function is based on the idea of calculating distances of "ideal" and "actual" conditions, in order to measure "how far" each proxy's *optimal* setting is from the *actual* technical setting. The decision for this approach was driven by the fact that both the service call records and the proxy characteristics are already in a vector-like form with ordered symbols ($s$, $m$, $h$) as values. This makes their distance-based comparison easy and meaningful. As this is done for each element of $R$, the results are then aggregated to a total score of each proxy for a given service. Thus, a utility function $n_{p_i}(r_j) \in \mathbb{R}$ must be formulated.

Let $R_{s_k} \subseteq R$ be the set of service call records of service $s_k$ and $N_{s_k,p_i}$ be the set of scores that result from the application of the utility function for $p_i$ upon $R_{s_k}$. Thus, $N_{s_k,p_i}$ contains the values of $n_{p_i}(r_j)$ for which $r_j \in R_{s_k}$. Let

$$N^+_{s_k,p_i} := \{x | x \in N_{s_k,p_i} \wedge x > 0\} \quad \text{and} \quad N^-_{s_k,p_i} := \{x | x \in N_{s_k,p_i} \wedge x \leqslant 0\}$$

then $\rho^+ := \frac{|N^+_{s_k,p_i}|}{|N_{s_k,p_i}|} = \frac{|N^+_{s_k,p_i}|}{|R_{s_k}|}$ is the quota of calls that would result in a positive score for this proxy.

Let $\gamma := [0..1]$ be the minimum acceptable threshold for $\rho^+$, $\beta := max\{0, \rho^+ - \gamma\}$ be the positive distance of $\rho^+$ to the threshold $\gamma$, and $\eta := \frac{\beta}{1-\gamma}$ be the ratio of this distance over the range of acceptable values for $\rho^+$. Then, the scoring function is

$$f(R,S,P,s_k,p_i) := \delta(\eta, |N^+_{s_k,p_i}|) \times \sum N^+_{s_k,p_i} + \delta(1-\eta, |N^-_{s_k,p_i}|) \times \sum N^-_{s_k,p_i}$$

with

$$\delta(a,b) = \begin{cases} 0, & \text{if } b = 0 \\ \frac{a}{b}, & \text{otherwise} \end{cases}$$

*Explanation*: The above description explains how results of single records are aggregated and $\gamma$ is only used for customization purposes. When the ratio of positive results is lower than $\gamma$, the positive results are ignored and only the negative ones are accumulated. If the ratio of positive results is between $\gamma$ and 100%, the values of positive and negative results are weighted and summed up as the result. For example,

$\gamma$ could be 0.6 for a minimum of 60% positive results, while $\gamma = 0$ means that no lower limit for the positive results is set.

Next, it must be defined how the utility function $n_{p_i}(r_j) \in \mathbb{R}$ works for single records. For each proxy, there is a description that consists of a set of conditions, i.e., thresholds (cf. Table 9). Let $\Psi$ be the set of attributes[3] $\{a_1, \ldots, a_{10}\}$ and $t_{p_i}(x)$ be the threshold of $p_i$ for the attribute $x \in \Psi$. For ease of use in the function to be definded, the symbols are mapped to integers with the mapping $z(v) := \{(s,1), (m,2), (h,3)\}$, maintaining their defined order. The symbol $u$ is a special case, describing the fact that the attribute value is unknown, hence a mapping to an integer is not necessary.

If $t_1$ is the operator ($\leqslant$, $\geqslant$, or $\emptyset$) and $t_2$ the value (s, m, h, or u) of a threshold t, while $y$ is an attribute value, then let

$$\varphi(y,t) := \begin{cases} 0, & \text{if } t_2 = u \\ \min\{\varphi(s,t), \varphi(h,t)\}, & \text{if } y = u \\ z(y) - z(t_2), & \text{if } t_1 = \geqslant \\ z(t_2) - z(y), & \text{if } t_1 = \leqslant \end{cases}$$

*Explanation*: The function $\varphi$ takes a tuple of a value (y) and a threshold (t) and calculates the distance between the two. This is done by mapping the values to numbers, and then calculating the difference. The difference must be positive if the value matches the threshold, negative otherwise. Positive differences have the meaning that the condition is satisfied. For example, an h value of an attribute where the condition is $\geqslant m$ gives a positive difference of +1. Although, for instance, a value m for a threshold $\geqslant m$ is also a *match*, the function $\varphi$ would give zero as a result. For this reason, all individual results of the function $\varphi$ that have constituted a *match*[4] will be later augmented by 1 by the utility function n. If the value (y) is u, the function $\varphi$ assumes that u could be any value of {s, m, h} and thus calculates the minimum difference, so that no positive proxy suggestions are made "by accident". In case the threshold is u, the difference is zero.

If for a record $r_j$ there is a match for all attributes, then its single score is the sum of the value to threshold distances of each attribute. Otherwise, the negative score is calculated as the sum of the value to threshold distances of parameter values not meeting the threshold.

Thus, the utility function is defined as

$$n_{p_i}(r_j) := \begin{cases} \frac{1}{\mu(p_i)} \times \sum_{x \in \Psi} 1_{\varphi(r_j(x), t_{p_i}(x)) < 0} \varphi(r_j(x), t_{p_i}(x)), \\ \qquad \text{if } \exists_{x \in \Psi} \varphi(r_j(x), t_{p_i}(x)) < 0 \\ 1 + \frac{1}{\mu(p_i)} \times \sum_{x \in \Psi} 1_{\varphi(r_j(x), t_{p_i}(x)) > 0} \varphi(r_j(x), t_{p_i}(x)), \\ \qquad \text{otherwise} \end{cases}$$

with

$$1_f := \begin{cases} 1, & \text{if } f = \mathit{true} \\ 0, & \text{otherwise} \end{cases}$$

3 The set of attributes is not denoted with A here, because $a_{11}$ is missing, so that $\Psi \neq A$.

4 These are all the positive results, but also all the "zero" results that have not been caused by an unknown threshold.

and

$$\mu(p_i) := \sum_{x \in \Psi} 1_{t_{p_i}(x) \neq u}$$

*Explanation*: If all thresholds are met, the proxy can certainly achieve benefits. This is the only case where the utility function assigns a positive value. Otherwise, it assigns a negative value. Obviously, proxies could offer benefits even when their score is zero or negative. Assigning positive values only to perfect matches is just one feature of the algorithm, which aims at giving positive values only when the benefit is a certainty. The resulting scores of the different possible proxies are probably going to be compared relatively, anyway. In all cases, the result is normalized by being divided through the number of values that are not u.

*Proof of the value range of the QoS-based decision support algorithm*

In the following, it is proven that the maximum score that can be assigned by the QoS-based scoring algorithm of Section 6.4 to a proxy is +3. The maximum score of function f is obtained when the second part of the expression $\delta(\eta, |N^+_{s_k,p_i}|) \times \sum N^+_{s_k,p_i} + \delta(1-\eta, |N^-_{s_k,p_i}|) \times \sum N^-_{s_k,p_i}$ is equal to zero, i.e., when there are no negative results, while the first part is maximized. In that case, the maximum score is:

$$\begin{aligned}
&\delta(\eta, |N^+_{s_k,p_i}|) \sum N^+_{s_k,p_i} \\
&= \frac{\max\{0, \rho^+ - \gamma\}}{1-\gamma} \frac{3|N^+_{s_k,p_i}|}{|N^+_{s_k,p_i}|} \quad \text{(because max } n_{p_i}(x) = 3) \\
&= \frac{\rho^+ - \gamma}{1-\gamma} 3 \\
&= \frac{\frac{|N^+_{s_k,p_i}|}{|N_{s_k,p_i}|} - \gamma}{1-\gamma} 3 \\
&= \frac{1-\gamma}{1-\gamma} 3 \quad \text{(if } N^+_{s_k,p_i} = N_{s_k,p_i}) \\
&= 1 * 3 \\
&= 3
\end{aligned}$$

Following the same principle, it can be proven that the minimum value is equal to -2, resulting in the value range [-2,3]. However, the scores that appear in reality for various different service call records usually are between -1 and 1, thus rendering the approach more easily comparable with probability-based scorings, such as this of the QoE-based algorithm of Subsection 6.4.2, which are per definition in the range [-1,1]. However, such a direct comparison has not been performed in the context of this thesis.

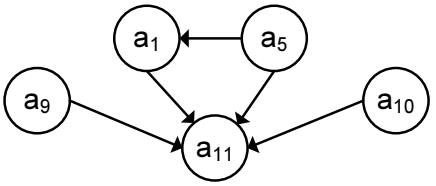
### 6.4.2 *Quality of Experience-based Scoring Algorithm*

Because users have their own subjective selection criteria, a different indicator of the suitability of the proxies is needed in this case. In QoE-based approaches, this indicator is based on user feedback, which can be explicit or implicit. The algorithm presented here uses past user decisions (i.e., user choices of proxies) as implicit feedback and calculates a "proxy suitability indicator" as its probability to be selected by the users in the future, if all proxies for this service exist. As the algorithm that calculates this probability should use part of the data in order to "learn" past user behavior and part of it in order to set evidence about the service that is examined each time, machine learning was an obvious choice. In particular, an algorithm based on a Bayesian Network (BN) has been developed, because BNs match the problem for two main reasons: First, they do not only classify cases (as simple decision trees do, for example), but they compute exact probabilities, as needed for a detailed proxy scoring. Second, they are an appropriate approach for setting evidences about future attribute values, which has to be done for every examined service.

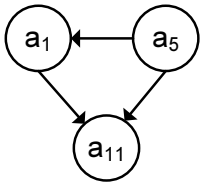
The idea is to let the algorithm learn about a given service by examining the past user behavior upon "similar" services which had been offered with all proxies. Two services $s_1$ and $s_2$ are similar if $a_i(s_1) = a_i(s_2), \forall i \in \{6,7,8\}$ (cf. Table 10), because $a_6, a_7$, and $a_8$ are the service-related attributes. The variables that are likely to determine the user selection are included in the BN, together with the variable about the user selection itself ($a_{11}$). These are the variables that are most probably known to the user, e.g., $\{a_1, a_5, a_9, a_{10}\}$. Summarizing in simple steps, the following is done in order to assign each proxy a score for a given service:

- *Step 1*: A logical BN structure is manually built, showing which attributes affect the user decision.
- *Step 2*: The records of similar services are analyzed in order to find out how users decide (generation of the Conditional Probability Tables of the BN).
- *Step 3*: The records of the examined service are analyzed in order to find out how the service is likely to be used in the future (Evidence in the BN).
- *Step 4*: Once the Conditional Probability Tables and the Evidences have been calculated, the BN is used in order to infer the probability of each proxy to be used for the examined service during the next calls. This probability is the final score of the proxy.

Figure 23 illustrates the aspects of the mentioned steps that are necessary in order to understand how the algorithm works. Concerning the BN structure (*Step 1*), manual construction is suggested instead of learning the structure from test data, because the causal relationships between the attributes are more or less straightforward. As [111] explains, in this case, the BN structure should be created manually by placing the causes before the effects. (*Step 2*) of Figure 23 shows example conditional probability tables for the variables used in a simplified version of the suggested BN. These Conditional Probability Tables are used together with the evidence (*Step 3*) in order to answer questions of the kind "what is the probability of a user selecting the proxy $p_x$ to invoke the service $s_k$?". Several tools and mechanisms exist [170] for the inference, i.e., the calculation of these probabilities (*Step 4*). Simple examples are presented

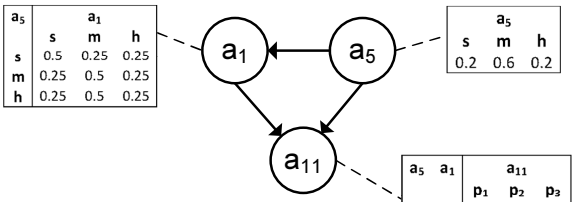



The suggested Bayesian Network structure: $a_{11}$ is influenced by $a_1$, $a_5$, $a_9$, and $a_{10}$, while $a_1$ is also related to $a_5$.




A simpler Bayesian Network structure to be used as an example for this illustration: $a_{11}$ is influenced only by $a_1$ and $a_5$, while $a_1$ and $a_5$ are, again, related.

***Step 1***




| $a_5$ | $a_1$ | | |
|---|---|---|---|
| | s | m | h |
| s | 0.5 | 0.25 | 0.25 |
| m | 0.25 | 0.5 | 0.25 |
| h | 0.25 | 0.5 | 0.25 |

| $a_5$ | | |
|---|---|---|
| s | m | h |
| 0.2 | 0.6 | 0.2 |

| $a_5$ | $a_1$ | $a_{11}$ | | |
|---|---|---|---|---|
| | | $p_1$ | $p_2$ | $p_3$ |
| s | s | 0.1 | 0.1 | 0.8 |
| s | m | 0.3 | 0.2 | 0.5 |
| s | h | 0.1 | 0.4 | 0.5 |
| m | s | 0.3 | 0.3 | 0.4 |
| m | m | 0.4 | 0.4 | 0.2 |
| m | h | 0.1 | 0.8 | 0.1 |
| h | s | 0.2 | 0.2 | 0.6 |
| h | m | 0.2 | 0.0 | 0.8 |
| h | h | 0.2 | 0.2 | 0.6 |

Example Conditional Probability Tables of the variables of the (simple) Bayesian Network of Step 1.

Here, for example, $P(a_{11}=p3 \mid a_5=h, a_1=m) = 0.8$.

These Tables are *learned* by analyzing the history of the services that are *similar* to the examined service

***Step 2***

$a_{11}$ is the *query variable*, while $a_1$ and $a_5$ are the *evidence variables*. At this step, *evidence* (E) is gathered for the evidence variables by analyzing the monitored service call records of the examined service.

For example, with the following four records for the examined service…

| | $a_1$ | $a_5$ | $a_{11}$ |
|---|---|---|---|
| $r_1$ | s | m | - |
| $r_2$ | h | h | - |
| $r_3$ | s | h | - |
| $r_4$ | s | s | - |

…the evidence state would be:
$e = (P(a_1=s)=0.75, P(a_1=m)=0, P(a_1=h)=0.25, P(a_5=s)=0.25, P(a_5=m)=0.25, P(a_5=h)=0.5)$

* Of course, evidence has to be gathered from a much higher number of records.

***Step 3***

Given the Conditional Probability Tables and the evidence state, beliefs for the probabilities of the query variable can be infered based on basic probability theory rules.

For example, $P(a_{11}=p_1 \mid E=e) = P(a_{11}=p_1, E=e) / P(E=e)$.
This would be then also the final score for $a_{11}$.

***Step 4***

Figure 23: Illustration of the core logic of the four steps of the QoE-based scoring algorithm

in [170], while [23] provides a detailed description of approaches for performing inference with soft, i.e., uncertain, evidence.

## 6.5 EVALUATION OF SELECTED IMPUTATION ALGORITHMS

As has been already mentioned –and will be explained in more detail in Subsection 6.5.1–, this evaluation section draws the focus upon the issue of missing data. The evaluation examines the efficiency of different imputation algorithms that can be used to enhance the application of the presented QoS-based algorithm for solving the ABS problem. The reasons for drawing the focus into that direction, along with further design decisions regarding the performed evaluation, are explained in the following subsection, before the description of the evaluation setup.

### 6.5.1 *Purpose and Setup*

For the case of handling the ABS problem with the presented QoS-based scoring algorithm, six state-of-the-art imputation algorithms are compared in six different test scenarios, in an attempt to investigate which of them achieves to minimize the error caused by missing data. The research potentials and the open issues concerning "missing data in the ABS problem" have been introduced in Section 6.3.

*General Decisions and Focus*

Because of the multitude of algorithmic solutions that may be applied, but also because the problem formulation (cf. Section 6.2) includes a big variety of factors, the ABS is a very broad issue, such that certain decisions concerning the focus of the evaluation are necessary in order to narrow the scope and reach interesting insights. Two important decisions have been made: (i) not to evaluate the quality of suggestions but the potential enhancements through the handling of missing data and (ii) to base the evaluation of imputation algorithms on the QoS-based scoring algorithm.

*The quality of the suggestions, i.e., the correctness of the delivered scores themselves, has not been in focus because*:

- The scores of the different scoring algorithms are based on different criteria. Different operators of the mediator layer (that would host the scoring algorithm) would, in turn, be interested on different factors, so that there is not a single algorithm to which all operators would "agree". In other words, there are currently no evaluation datasets, along with corresponding optimal scoring outputs, which would not be highly objective. Exactly because scores are up to an extent subjective, they are used as decision support suggestions and not as automated action triggers. As already mentioned, it is assumed that the scores of the previously presented scoring algorithms are meaningful because the scoring algorithms are based on the insights of an educated survey. Potential users of the algorithms (i.e., mediation layer operators) should evaluate the quality of the algorithms based on their own systems and their own expert opinions.
- From a scientific point of view, the comparison of imputation algorithms seems to be more interesting than the comparison of case-specific scoring outputs

because of three reasons: Firstly, the evaluation methodology (or specific parts of the evaluation) can be reused in order to evaluate the efficiency of imputation algorithms in other related scenarios. Secondly, in the examined domain, missing monitored data exist with such a high probability (cf. also [54]) that the errors caused by them may be much more significant than the "errors" caused by imperfections of the scoring logic. Thirdly, as already implied above, the evaluation related to missing data can be performed more objectively.

For the evaluation of the missing data handling, i.e., for the comparison of imputation algorithms, *the QoS-based scoring algorithm has been preferred*[5] *over the QoE-based scoring algorithm because*:

- It is *more representative* for handling the ABS problem, because it considers all context attributes (and not only those that may affect the user's choice), because it is more directly based on the survey results and more intuitively derived from them, and because QoS-approaches are in general less subjective and probably more widely used in the domain of networks.
- It uses a straightforward logic and only a few steps for the calculation of the scores. Thus, it seems to introduces *less "noise"* than the QoE-based algorithm (or similar algorithms) with regard to the effect of missing data. For example, as shown in the experimental results of [103], the effect of missing data in the case of the QoE-based algorithm may depend heavily on the number of the available Web service call records (elements of the set R, cf. Section 6.2), because a minimum amount of records is necessary in order to "learn" the user behavior efficiently. However, this is a characteristic of machine-learning algorithms which is not considered to be ABS-specific, so that any related effects are not especially interesting and should be avoided or filtered. [103] also shows that the effect of missing data in the case of the QoS-based algorithm does not depend on this factor.

Decisions about the imputation algorithms selected for comparison, the test datasets, the examined cases of missingness, and the used metrics are explained in the paragraphs that follow.

*Compared Imputation Algorithms*

The algorithms that have been examined are those that have been listed as state-of-the-art imputation algorithms in Subsection 2.3.3. These algorithms cover the different possible classes of algorithms and are often used in similar surveys [87]. Together with them, the case of no imputation at all is included as a sixth approach, meant to be the baseline with which the results of the other algorithms are compared. It is, of course, expected that the case of no imputation will give the worst results, but this should not be taken for granted, because in extreme cases, the "guesses" about missing data can be worse than not guessing at all. The compared approaches are:

- No Imputation

5 However, some relevant evaluation results for both the QoS- and the QoE-based scoring algorithm have been presented in [103].

- Case Deletion
- Modus Imputation
- Random Hot Deck
- Distance Function Matching
- Multiple Imputation

More information about the algorithms can be found in Subsection 2.3.3 and in the respective referenced publications, e.g., [87].

*Test Scenarios*

The current evaluation considers six scenarios for building the test datasets. More concretely, two ways for generating complete test data (without unknown values) and three ways for "inserting" unknown values into them have been used, resulting in six different combinations concerning the way the final datasets are generated. These combinations are called test scenarios.

The *data* themselves, before the insertion of missingness, have been generated in the following two ways:

- *Random*: All values of the attributes of the Web service call records are generated randomly.
- *Scenario-based*: Scenario-related assumptions are used for data generation, so that randomly generated attribute values have an effect on the probabilities of particular values for other attributes. More concretely, two assumptions are made. Firstly, it is assumed that certain services are likely to be consumed by the same clients or by clients with similar characteristics. Secondly, it is assumed that a relationship between device capabilities and used network connection exists, i.e., that users with more capable devices tend to use better connections, both because their devices can technically achieve it and because they can afford it.

The *missingness* is then inserted into the generated complete data sets with one of the three following methods[6]:

- *Random*: Randomly selected values of the complete data set are marked as unknown (u).
- *Unreliable data sources*: The insertion of unknown values is based on an analysis of the sources that normally monitor or collect the respective data. The attributes are divided into those that can be measured by the mediation layer and those that are potentially known only by the service consumer itself. The first class contains the attributes of the Web service, as well as the latency and the packet loss of the connection during a Web service call. The rest (bandwidth, stability,

6 The exact probabilities and correlations have been chosen in a way that all cases end up having a missingness of ca. 25%, in order to obtain comparable results. The exact values are not critical, as it should first be examined if the different scenarios affect the efficiency of the algorithms at all.

Table 12: The six test scenarios for the evaluation of imputation algorithms in "Always Best Served"

| TEST SCENARIO | DATA | MISSINGNESS |
|---|---|---|
| RR | Random | Random |
| RS | Random | Unreliable data sources |
| RC | Random | Unreliable data collection |
| SR | Scenario-based | Random |
| SS | Scenario-based | Unreliable data sources |
| SC | Scenario-based | Unreliable data collection |

CPU power, service call frequency, service call criticality) should be reported by the consumer. The latter have a higher probability to be missing, while the attributes latency and packet loss, which can be monitored by the mediation layer, tend to be missing together, when they are missing.

- *Unreliable data collection*: The previously described way of inserting missingness is extended here by taking into account not only the sources of the data, but also their collection and transmission. Assuming that the attributes that are monitored on the device have to be transmitted to the mediation layer at some point, capable devices are considered to have a higher probability of performing or achieving this. Thus, records with a value $s$ for CPU power have a higher probability of missing the rest of the "on device-measured" attributes.

It is noted that the first two cases are different cases of MCAR missingness (cf. Section 2.3), while the third case is MAR, because the missingness of attributes does not only depend on luck (as in the first case) or on the missingness of other attributes (as in the second case), but also on attribute values, namely on the value of CPU power. The six different test scenarios are summarized in Table 12.

*Metric*

The metric used for the comparison of the examined imputation algorithms is the error imposed by the missing values on the result of the scoring algorithm. It will be examined how the application of the imputation algorithms reduces this error. As there are different ways for defining error metrics, a series of formal descriptions is provided, not only in order to mathematically define the used error metric, but also in order to provide a better understanding of the steps of the conducted measurements. References are also made to the symbols and definitions of Section 6.2, including the sizes of the sets defined there.

If a *dataset* $D \in V^{|R|\times|A|}$ is a matrix that contains the values of the attributes of the monitored Web service call records, G is the set of the (two) data generation scenarios (cf. previous paragraph), M is the set of the (three) missingness scenarios (cf. previous paragraph), H is the set of the (six) examined imputation algorithms, and I is the set of the performed repetitions of the experiment, then:

- The set of the *test cases*, each of which is going to be repeated $|I|$ times, is:

$$T = G \times M \times H$$

- For $i \in I$ and $t \in T$:

$$D_{i,t} : I \times T \longmapsto V^{|R|\times|A|}$$

is the dataset resulting for the $i$-th iteration of the test case $t$. *For each* such dataset, a reference dataset exists, which is used for the calculation of the error. This reference dataset refers to the respective case without missingness and without imputation, such that it is denoted as:

$$D_{i,b_t} : I \times G \longmapsto V^{|R|\times|A|}$$

where $b_t$ simply denotes the reference test case of the test case $t$.

- Thus, safely abstracting from any other information used by the scoring algorithm, the scoring function ($f$) can be defined as a function that maps a dataset to a matrix of scores for the service-proxy pairs (remember that $B$ is the range of scores that can be assigned):

$$f(D) : V^{|R|\times|A|} \longmapsto (B \subset \mathbb{R})^{|S|\times|P|}$$

- The error ($e$) of a scoring output is calculated as the difference between this scoring output and the scoring output for the respective reference test case and this error is calculated for all test cases. Thus:

$$e(D_{i,t}) = |f(D_{i,b_t}) - f(D_{i,t})| \quad ,$$
$$e(D_{i,t}) : V^{|R|\times|A|} \times V^{|R|\times|A|} \longmapsto [0, b_{max} - b_{min}]^{|S|\times|P|}$$

- The final metric is the above error normalized by the observed[7] range of its possible values and calculated as percentage of this range. This metric represents better the extent (in percent) to which the missing data affect the scoring output and is the variable that will be plotted in the results of Subsection 6.5.2. Thus, the final metric is:

$$e_{norm}(D_{i,t}) := \frac{e(D_{i,t})}{b'_{max} - b'_{min}} \times 100\%$$

The average values and the standard deviation of $e_{norm}$ can be calculated and used for the comparison of the imputation algorithms. Different error metrics could have been used, e.g., comparing the changes in the ranking of proxies caused by missing data. However, the scores are supposed to be suggestions. Therefore, the errors of every individual value are equally valued, which leads intuitively to the use of the calculation of differences. The errors could be further grouped "per service", e.g., by using the euclidean distance of the two respective vectors when comparing the results for two services. This is performed in [103] because the results are presented per service. Here, however, the focus is just on the behavior of the imputation algorithms as a whole, service-independently.

7 Note that not the theoretical range of scores, but the one observed throughout the experiment is used, so that $b'_{max}$ and $b'_{min}$ may differ from $b_{max}$ and $b_{min}$.

*Further Details*

The measurements have been conducted with a software which has been realized to execute the necessary steps and also store intermediate results in order to support a better tracking of the process and further analyses of the results.

The main software program has been implemented with the Java programming language, while for the tests of the Multiple Imputation algorithm, the external statistics program R[8] has been integrated. Scripts that call the R program are used in order to write intermediary results into files which are then read by the main Java program. The Multiple Imputation implementation that is used in R is described in [162].

Concerning the length of the datasets, namely $|R|$, [103] indicates that it does not affect the results. Thus, the experiments have been performed with $|R| = 10000$, as this size is big enough for eliminating effects of luck in single repetitions and can still be processed in reasonable time. Similar is true for the amount of services that appear in the records (six have been used), while the rest of the information needed by the QoS-based algorithm (proxy characteristics) is taken directly from the survey results of Section 4.4 (four proxies from the corresponding table have been used). These variables are not in focus, because individual tests (cf. also [103]) indicate that they do not play a decisive role with regard to the results presented in the next subsection.

Each test case has been repeated ten times (i.e., $|I| = 10$). This number of repetitions has been enough in order to achieve sufficiently small confidence intervals for the average errors, as will be seen in the results.

### 6.5.2 *On the Comparison of Imputation Algorithms*

The results, i.e., the average values and the corresponding (99%) confidence intervals[9] of $e_{norm}$ for the examined imputation algorithms in the six different test scenarios (cf. Table 12) are presented in the six corresponding graphs (Figure 24 to Figure 29). The observations that can be made based on these results do not only relate to the efficiency of the examined imputation algorithms, but also to the meaning of the standard deviations and the confidence intervals that appeared in the results and to the meaning of the differences (or similarities) between the results of the different test scenarios. Finally, some comments about the complexity of the imputation algorithms are provided, although this aspect is normally not critical in the examined scenario and no detailed measurements of execution times are presented here. This is because proxy generations are performed a priori offline and not real-time during particular Web service calls.

Three of the imputation algorithms, namely *Case Deletion, Random Hot Deck, and Multiple Imputation, deliver the best overall results*, minimizing the error caused by missing data in the scoring output. This is common in all the results and the statement cannot be affected by the deviations that appear, because the named algorithms present in *all* cases an error of under 2.5%, which is comparable only with best-case errors (outliers) of the other approaches, while the average errors of the latter are usually many times bigger. Among the best approaches, Multiple Imputation seems to have the smallest average error, but without statistically significant differences (i.e.,

8 `http://www.r-project.org` (Last accessed in January 2012).

9 cf. Section A.3 about the calculation of the confidence intervals.

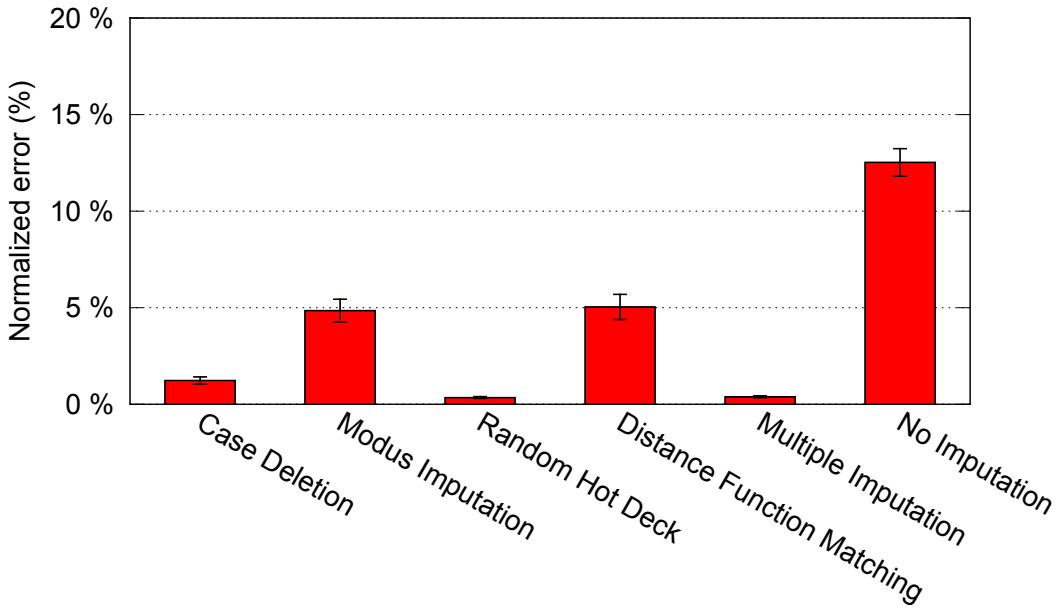



Figure 24: Average normalized error of the imputation algorithms in the test scenario RR (Random data, random missingness)

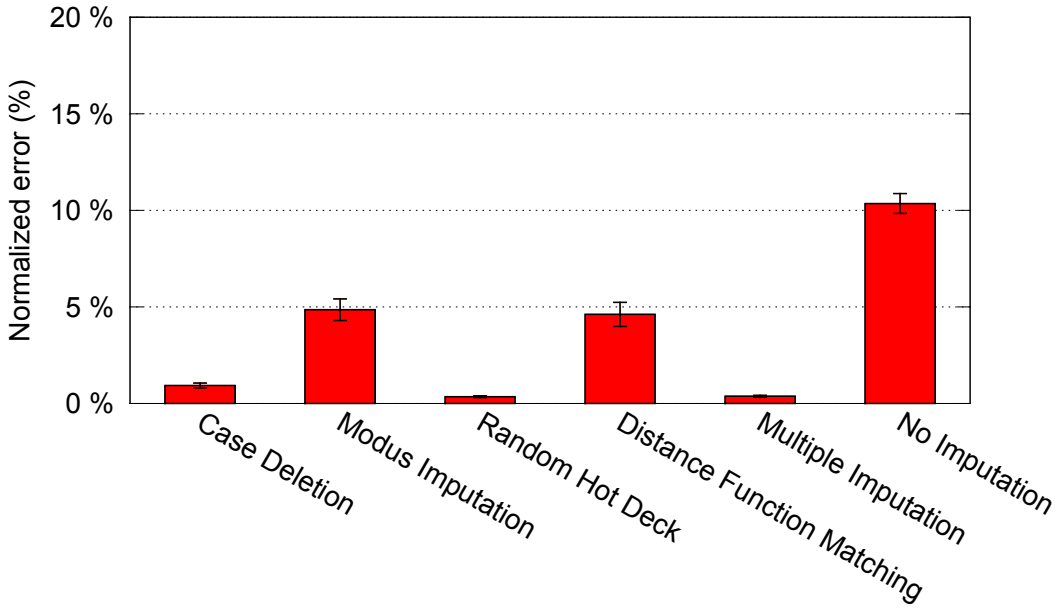



Figure 25: Average normalized error of the imputation algorithms in the test scenario RS (Random data, missingness from unreliable data sources)

not always and with overlapping confidence intervals). Reminding that the results are used for decision support, the errors of all three best algorithms are in any case so small that normally the choice should be between Case Deletion and Random Hot Deck, in order to avoid the (implementation- and time-) complexity of Multiple Imputation. More information with regard to complexity is provided later. Finally, as

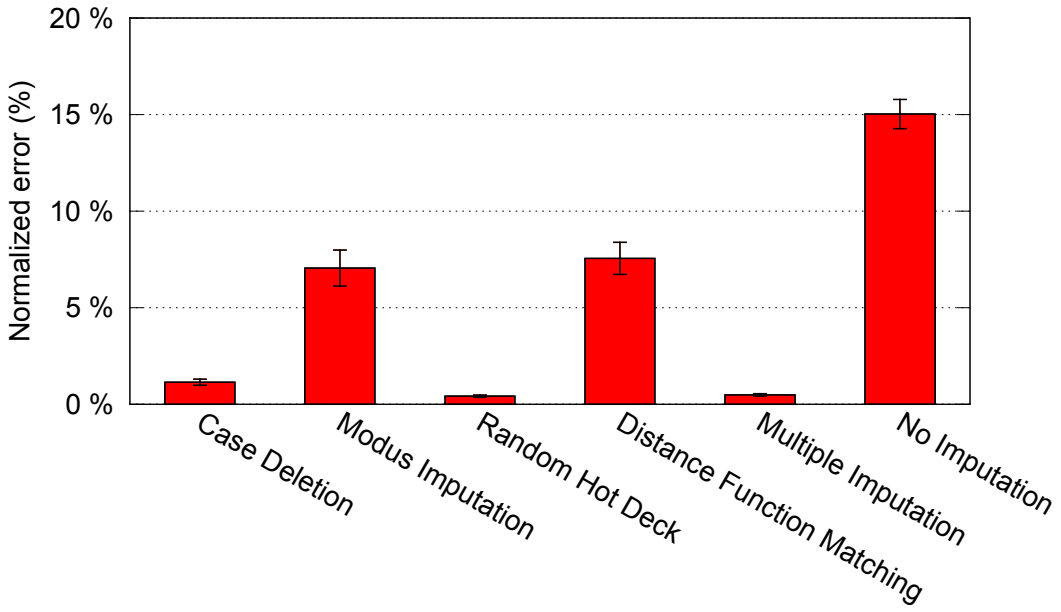



Figure 26: Average normalized error of the imputation algorithms in the test scenario RC (Random data, missingness from unreliable data collection)

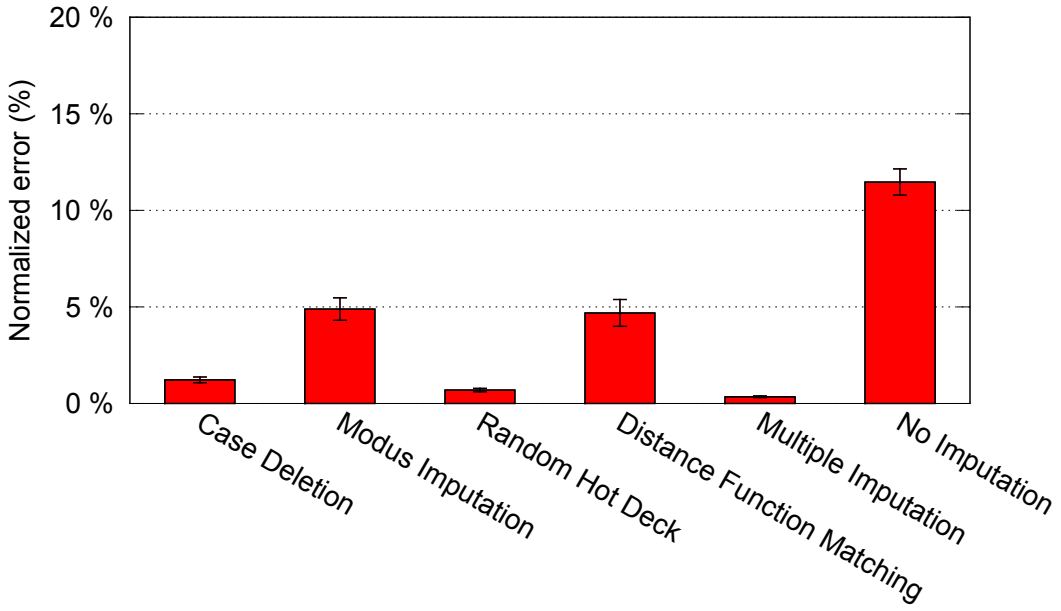



Figure 27: Average normalized error of the imputation algorithms in the test scenario SR (Scenario-based data, random missingness)

expected, the complete absence of imputation leads to very big errors in the scoring output, usually bigger than 10%.

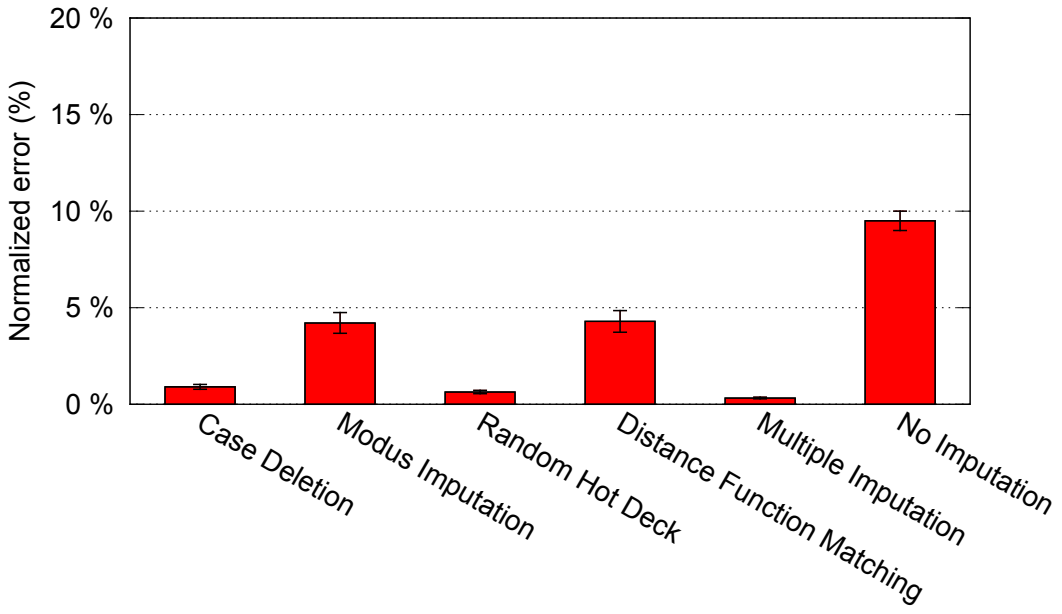



Figure 28: Average normalized error of the imputation algorithms in the test scenario SS (Scenario-based data, missingness from unreliable data sources)

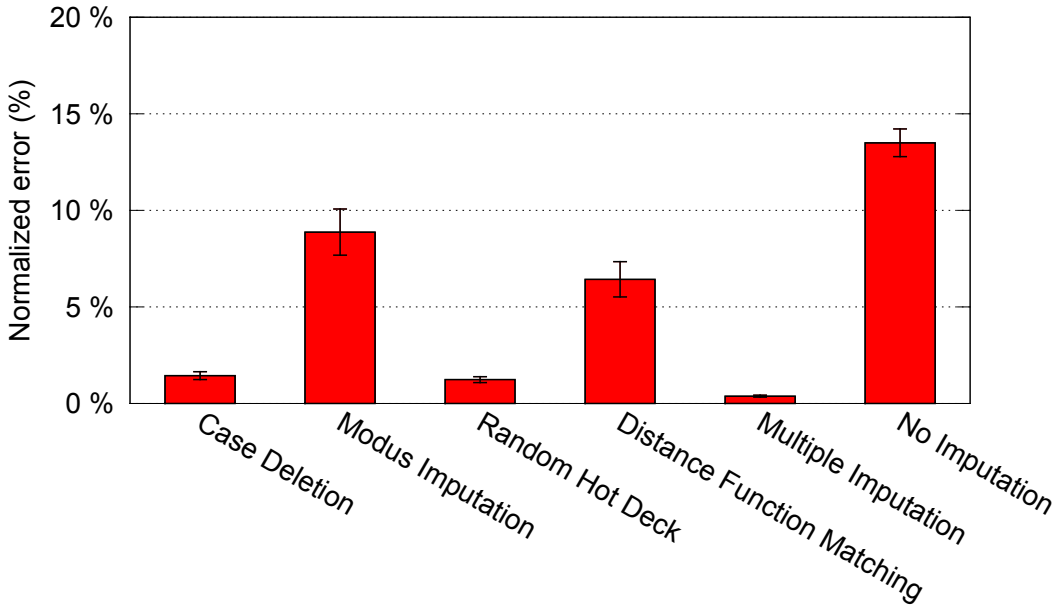



Figure 29: Average normalized error of the imputation algorithms in the test scenario SC (Scenario-based data, missingness from unreliable data collection)

### 6.5.3 *Discussion*

An interesting observation regarding the results per test scenario is that, in general, *no significant differences in the efficiency of the imputation algorithms can be observed for the different test scenarios*. Only in the two scenarios with data collection-related missingness (RC, SC) seems the difference between the good-performing and the

bad-performing algorithms to become bigger, which is probably because of the MAR missingness and the fact that the missing data of these test scenarios cause a bigger "initial" error (cf. the results for "No Imputation"). However, it should be mentioned that the good-performing algorithms achieve for these test scenarios very similar errors as for the other test scenarios. This similarity between the results leads to a very important conclusion: The differences in the efficiency of the imputation algorithms lie rather in the nature of the problem (use of discrete categorical values, use of survey results for the proxy characteristics, scoring algorithm logic) than in the distribution of the values and the types of missingness.

An observation that has been made in parallel to the analysis of the overall efficiency of the imputation algorithms is that *relatively big standard deviations appeared in the results*. However, this cannot be seen on the graphs[10], as enough repetitions have been performed in order to achieve a sufficient confidence about the average errors. Only overlapping confidence intervals could reduce the confidence in the comparative statements. Obviously, the facts mentioned in the previous paragraph concerning the three best approaches and the inefficiency of the absence of imputation cannot be affected by the standard deviations. The standard deviations seem to be caused by the fact that, among others, results of different proxies are summed and averaged. The fact that some of the proxies are scored very negative or very positive because of their restrictions, results in that their scores are not really affected by the missing data, i.e., they remain "very good" or "very bad" even when data is missing. Thus, for such proxies, no imputation algorithm presents big errors. This aspect could be further investigated, but it is here out of scope, because big errors for "only" some of the proxies are enough to affect the decision process.

As expected because of the different complexities of the algorithms, *some of the imputation algorithms have been much more time consuming than the others*. Although many statistical software packages (such as R, which has been used in the experiments) provide practical and relatively efficient implementations of complex imputation algorithms, some of them are computationally intensive and time consuming [24]. Indeed, while most of the algorithms completed their work in seconds, Multiple Imputation and Distance Function Matching often needed many minutes for a single iteration. As already mentioned in Section 6.3, the decisions are taken offline whenever the operator needs to decide which proxies to generate and not on-the-fly at the moment of particular Web service calls. Therefore, complexity may not be critical and is out of scope here. However, given the very similar results of Multiple Imputation with, e.g., Case Deletion, it is very probable that the first would not come into question when dealing with big datasets (as in the experiment), because so long execution times are often undesirable or disturbing even if the process is performed offline.

All in all, with retrospect to the part of the research question that refers to finding the best way to make the scoring of Web service proxies resistant to missing context data, the experiments have provided valuable results, which indicate the adequacy (or not) of the imputation algorithms. Although decision makers may pose different requirements on the decision support system, Case Deletion and Random Hot Deck should cover almost all cases with satisfactory results and small implementation effort and computation overhead. Unless the data types, the value ranges, the used algorithms, and the proxy descriptions are dramatically changed, one of these two

10 cf. Section A.3 about the calculated standard deviations.

imputation approaches should be probably applied for enhancing the scoring of Web service proxies when data is missing. Even in the case of a heavily changed problem, a detailed examination of the results and the analysis of the previous subsection should help identify if the changes in the nature of the problem should lead to considering other solutions.

## 6.6 SUMMARY AND CONCLUSIONS

This chapter has focused on the case in which a mediation layer has to choose among different possible Web service adaptation mechanisms, i.e., among (some of) the different proxies that have been studied in the previous chapters. Accordingly, the corresponding decision problem has been mathematically formulated, two scoring algorithms have been presented for its solution, and six imputation algorithms have been tested and evaluated with regard to their adequacy for making the decisions resistant to missing context data.

The formulation of the decision problem has narrowed the field of study by setting the requirements of a Web service proxy scoring algorithm for decision support. Then, the exploration of related work has explained, among others, why a case-specific analysis is necessary in order to find out which imputation algorithms would perform better in the examined scenario.

Next, two scoring algorithms have been proposed for the solution of the formulated decision problem. They represent the two big classes of solutions in decision support, namely QoS-based and QoE-based approaches. The scoring results themselves have not been evaluated due to lack of objective reference data sets, but also because they are assumed to be "appropriate enough", as they are based on the insights of the corresponding survey of Chapter 4.

Instead, an evaluation of imputation algorithms that can be employed for the handling of missing context data in the discussed problem has been performed. The evaluation results have shown the potential for enhancing the decision support significantly by using imputation algorithms. Case Deletion, Random Hot Deck, and Multiple Imputation have been shown to be the most promising imputation algorithms for maintaining the quality of the proxy scoring results in a scenario with missing context data. Further, the evaluation has also shown that striving towards the development of a new, scenario-tailored imputation algorithm is not a worthwhile goal, unless extreme requirements for the accuracy of the scoring output are set. This is because, with loose requirements, the results of Case Deletion, Random Hot Deck, and Multiple Imputation cannot be enhanced much more.

# 7

# CONCLUSION AND OUTLOOK

»Begin thus from the first act, and proceed; and, in conclusion, at the ill which thou hast done, be troubled, and rejoice for the good.«

— Pythagoras

Concluding the thesis, this chapter provides some final remarks along with a summary of contributions, as well as a discussion of the possibilities to exploit the research results or extend the work towards future research directions.

## 7.1 CONCLUSION

SOC and mobile computing are two successful disciplines that originated from very different domains and have grown in parallel. It is certain that it has often been difficult to let the powerful, verbose, self-descriptive messaging technologies of SOC converge with the "thin", lightweight nature of the processing and communication modules of mobile devices. However, the huge potentials could not be ignored. Instead, ways had to be found for allowing the (wireless) mobile devices to participate in service-oriented systems without overburdening their limited resources.

In the field of the most important SOC-related technology, namely standard XML-based Web services, the reaction to the aforementioned issue appeared in the form of Web service adaptation mechanisms, i.e., approaches which attempt to reduce the load caused by the Web service invocations on the device, usually by attempting to minimize the wirelessly transmitted and on-device-processed data. In addition to making communication more lightweight for wireless mobile participants, adaptation mechanisms have also been employed in order to enhance further aspects, e.g., the automated context-enrichment or the handling of technical incompatibilities of Web service invocations that are performed by wireless mobile clients.

Although this thesis discussed different types of adaptation mechanisms and presented an architecture and a framework for Web service adaptation, called Mobility Mediation Layer, the focus has been laid on adaptation mechanisms related to performance. In this context, a survey and analysis of possible (performance-related) adaptation mechanisms has been presented in order to identify open issues and to lay a foundation for further research on this issue. Then, a new Web service adaptation mechanism has been developed and, finally, the issue of appropriately choosing the most adequate adaptation mechanisms under changing conditions of the mobile system context has been elaborated.

While carrying out these research activities, various contributions have been achieved. Although the contributions have already been abstractly listed in the Introduction, they are revisited in the following, as all technical details and the results of the corresponding work have now been explained to the reader. In accordance with the mentioned research activities, three major contributions can be identified,

while secondary contributions stem from partial results, collateral experiments, or theoretical analyses that have been necessary in fulfilling the research goals.

The first main contribution is the result of the survey of Web service adaptation mechanisms that can enhance the performance of mobile wireless Web service consumption. A case study helped to identify four network-related aspects (bandwidth, latency, packet loss, stability), one device-related aspect (CPU power), three service-related aspects ("Data size"-to-"Web service message size" ratio, Web service message size, processing time) and two application-related aspects (service call frequency, service call criticality) as the aspects of the system context that should be considered in the analysis. Then, eleven approaches (based on fourteen publications) have been analyzed with regard to the conditions of the system context that render them beneficial. The survey has been complemented by experiments exclusive for this thesis, for which no similar experiments have been found in literature. The results of the survey indicate that different adaptation mechanisms make sense under different conditions. For example, on-the-fly compression of Web service responses is more beneficial when the response sizes are large, while SOAP-to-RMI protocol transformation makes sense either when the response sizes are small or when the responses include complex data types. These survey results can serve as the "heart" of corresponding decision support systems, which could be part of mediation layers, assisting system designers and developers to decide what kinds of proxies should be generated and used for certain Web services. Further, the survey results assist further research by helping to identify open issues and potentials for the development of new performance-oriented Web service adaptation mechanisms.

The second main contribution is the development of a new, caching-based Web service adaptation mechanism. The developed approach enables the automated and generic generation of Web service proxies that enrich the communication with external, possibly third-party Web services, with the following capability: Responses of the Web services can be cached and used on (mobile) consumer devices with guaranteed freshness, while the complete responses are re-transmitted only when changes have occurred. This capability is known from the WWW and the caching of Web content, and its absence in the world of Web services has been known and discussed, implicitly or explicitly, in related prominent surveys and technical documents [43, 159]. The innovative proxy generation process has been described with sequence diagrams and pseudocode extracts. The enabled communication scheme has been evaluated concerning the saved bandwidth and the reduction of (mobile) user-perceived latency in different scenarios. It is characteristic that the benefit of the approach does not have upper limits, as it may spare the transmission of messages that can theoretically be infinitely big. However, the benefit of the approach depends on the service call traces, i.e., on the frequency of identical service calls, the frequency of changes in the responses etc. Therefore, detailed evaluation results have been presented for service call traces that have been accounted to be realistic, again indicating the significant benefits of the approach.

The third main contribution is the analysis of decision support algorithms for Web service mediation layers, so that the latter can employ the most adequate Web service adaptation mechanisms under given conditions of the mobile Web service usage. After formally describing the problem and after proposing two scoring algorithms (one QoS-based and one QoE-based) that are based on the insights of the related survey (cf. first contribution), the focus has been laid on the scenario-related handling of missing

data. Based on the results of an evaluation which has been performed for the first time for this kind of problem, two main findings concerning the handling of missing data for the scoring of Web service adaptation proxies have been discussed: Firstly, the simple imputation approaches of Case Deletion and Random Hot Deck should be preferred unless the problem formulation and its details are heavily changed. Secondly, the investigation for the development of new imputation algorithms, which would be tailored to this particular scenario, is not a worthwhile pursuit.

Secondary contributions of this thesis can be found in the modeling of the adaptation lifecycle for mobile Web service usage, in the architecture/framework for the mediation of mobile Web service calls, in the survey of Web service response caching approaches, in the development of particular scoring algorithms, as well as in the examination of the subject of Web service adaptation for wireless participants as a whole. The latter has helped to identify research challenges and open issues, but also important engineering aspects.

## 7.2 OUTLOOK

Naturally, each detail of a research approach –each algorithm and each design decision– may be open to modifications or extensions, with the possibility of enhancements. For example, with regard to the presented caching-based adaptation mechanism, it could be investigated if extensions that enable the use of cached data even in cases of cache misses could also be implemented generically and enhance performance. The idea would be to automatically generate proxies that can always calculate the "XML-difference" of the current Web service response with the one in cache and transmit only this difference under certain circumstances. Further bandwidth savings would be the advantage of this approach. Some disadvantages could be the implementation complexity and the necessity to include XML-difference handling logic at the client-side, as well as the processing overhead that comes along with it. Other example extensions can be thought of, as well.

However, regarding the research conducted in this thesis as a complete and thoroughly evaluated work, it is more interesting –and more helpful for future researchers– to examine how the insights gained herein can be exploited to set new research goals and build the foundation for examining further aspects, always considering the ever-changing needs of the researched technology area.

Therefore, given the importance of energy or battery consumption in modern and future systems [112, 113], but also the evolution of new Web-based technologies, especially for being used in the Cloud [56], two corresponding parts of the work at hand can be further researched in order to develop innovative solutions in the respective directions. Firstly, how the situation changes if the focus of Web service adaptation is drawn away from performance and towards energy can be further researched. Secondly, if and how far it makes sense to develop similar adaptation mechanisms for related Web-based technologies other than standard XML-based Web services can also be researched.

*With regard to the development of energy-aware Web service adaptation mechanisms*, the critical point lies in the fact that the majority of Web service adaptation mechanisms (including those discussed and developed in the context of this thesis) focus heavily on the reduction of the data that is processed on and wirelessly transmitted by

the device. Concerning performance, usually measured as response times or user-perceived latency (response plus parsing and processing times), this has been an overwhelming factor. Although the wirelessly transmitted data is a dominating factor of energy consumption, it has not yet been thoroughly examined whether other service-related factors affect the results in a similar way, or even more heavily. Generally, arguments such as "the data transmitted wirelessly during the Web service invocation have been reduced, thus energy is being saved" are withdrawn, as factors like the number of connection establishments to the server or the total time spent by a device outside the "idle" status have proven to be sometimes dominating over the amount of wirelessly transmitted data. Therefore, a researcher should begin with experiments that reveal the impact of service-related characteristics (e.g., length of time intervals between service invocations, importance of connection establishments compared to data transmission etc.) on energy consumption. Then, corresponding rules and models could be extracted, which would in turn inspire the development of energy-aware Web service adaptation mechanisms.

*With regard to adjusting Web service adaptation mechanisms to related modern Web-based technologies* (which exist or will evolve), the questions are whether or not the corresponding messaging mechanisms include self-description overhead in a similar way to standard Web services and whether or not particular mechanisms for reducing the load of messaging (such as validity-check-enabled caching) are inherently (or can easily be implemented as) part of these technologies. For example, it would probably not be very promising to develop generic and automated mechanisms for performing translation of REST messages into another format, protocol, or representation (because the messaging overhead of the alternative format is not likely to be much smaller), but it would probably be worthwhile to investigate what other techniques can be employed in order to crop data intelligently. Further, the intensive research and the fast advances in the Cloud domain may result in new technologies for Web-based service provision and consumption. Thus, these advances should be closely observed, in order to come up with solutions for adapting these technologies to the needs of wireless mobile devices. For example, new ideas are emerging in research communities with regard to the participation of mobile devices in the Cloud, not only as service users, but also as participants that offer resources. Such a technical landscape could further complicate the nature of the description- and messaging-overhead utilized, so that the applicability of adaptation mechanisms such as those examined in this thesis should be re-examined and provided through corresponding extensions.

All things considered, SOC technologies are still far from being completely harmonized with the features of mobile computing. However, viewing the work at hand as a step towards achieving this goal, researchers could continue on making service-based communication of the future really seamless. This should be pursued not only with regard to technological complexity, which is often already well hidden, but also with regard to performance. Towards this goal, it should not be forgotten that the devices of future service-oriented systems will not necessarily look like today's devices. Although –fortunately– no one can tell the future, no technological advancement is likely to eliminate all other technologies and, with that, the benefits of adaptation.

# BIBLIOGRAPHY


[1] Osama Abboud, Konstantin Pussep, Aleksandra Kovacevic, and Ralf Steinmetz. Quality Adaptive Peer-to-Peer Streaming using Scalable Video Coding. *IFIP/IEEE International Conference on Management of Multimedia and Mobile Networks and Services (MMNS '09)*, pages 41–54, 2009.

[2] Mustafa Adacal and Ayse Bener. Mobile Web Services: A New Agent-based Framework. *IEEE Internet Computing*, 10(3):58–65, 2006.

[3] Erwin Aitenbichler, Jussi Kangasharju, and Max Mühlhäuser. MundoCore: A Light-weight Infrastructure for Pervasive Computing. *Pervasive and Mobile Computing*, 3:332–361, 2007.

[4] Paul Allison. *Missing Data Thousand Oaks*. SAGE Publications, 2001.

[5] Mehmet Altinel, Christof Bornhövd, Sailesh Krishnamurthy, Chandrasekaran Mohan, Hamid Pirahesh, and Berthold Reinwald. Cache Tables: Paving the Way for an Adaptive Database Cache. In *International Conference on Very Large Data Bases (VLDB '03)*, pages 718–729. VLDB Endowment, 2003.

[6] Android Developers. Android Emulator Documentation. URL `http://developer.android.com/guide/developing/devices/emulator.html`. Last accessed in January 2012.

[7] Naresh Apte, Keith Deutsch, and Ravi Jain. Wireless SOAP: Optimizations for Mobile Wireless Web Services. In *Special Interest Tracks and Posters of the International Conference on World Wide Web (WWW 2005)*, pages 1178–1179. ACM, 2005.

[8] Stefan Arbanowski, Pieter Ballon, Klaus David, Olaf Droegehorn, Henk Eertink, Wolfgang Kellerer, Herma van Kranenburg, Kimmo Raatikainen, and Radu Popescu-Zeletin. I-centric Communications: Personalization, Ambient Awareness, and Adaptability for Future Mobile Services. *IEEE Communications Magazine*, 42(9):63–69, 2004.

[9] Michael Armbrust, Armando Fox, Rean Griffith, Anthony D. Joseph, Randy H. Katz, Andrew Konwinski, Gunho Lee, David Patterson, Ariel Rabkin, Ion Stoica, and Matei Zaharia. Above the Clouds: A Berkeley View of Cloud Computing. Technical Report UCB/EECS-2009-28, EECS Department, University of California, Berkeley, 2009. URL `http://www.eecs.berkeley.edu/Pubs/TechRpts/2009/EECS-2009-28.html`. Last accessed in January 2012.

[10] Ivan Avdouevski, Riitta Kerminen, Jarmo Makkonen, and Ari Visa. Missing Values in User Activity Classification Applications Utilizing Wireless Sensors. In *ACM International Symposium on Mobility Management and Wireless Access (MobiWac '08)*, pages 151–155. ACM, 2008.

[11] Niranjan Balasubramanian, Aruna Balasubramanian, and Arun Venkataramani. Energy Consumption in Mobile Phones: a Measurement Study and Implications for Network Applications. In *ACM SIGCOMM Internet Measurement Conference (IMC '09)*, pages 280–293. ACM, 2009.

[12] Greg Barish and Katia Obraczke. World Wide Web Caching: Trends and Techniques. *IEEE Communications Magazine*, 38(5):178–184, 2000.

[13] Rainer Berbner. *Dienstgüteunterstützung für Service-orientierte Workflows*. Books on Demand, 2008.

[14] Peter Brittenham, Rob Cutlip, Christine Draper, Brent Miller, Samar Choudhary, and Marcelo Perazolo. IT Service Management Architecture and Autonomic Computing. *IBM Systems Journal*, 46(3):565–581, 2007.

[15] Thomas Buchholz and Claudia Linnhoff-Popien. Towards Realizing Global Scalability in Context-Aware Systems. In *Location- and Context-Awareness*, pages 26–39. Springer, 2005.

[16] Arndt Buschmann, Christian Werner, Stefan Schmidt, and Stefan Fischer. Protokollunterstützung für SOAP Web Services auf Mobilen Geräten. *Praxis der Informationsverarbeitung und Kommunikation*, 27(3):140–144, 2004.

[17] Claudia Canali, Michele Colajanni, and Riccardo Lancellotti. Performance Evolution of Mobile Web-based Services. *IEEE Internet Computing*, 13(2):60–68, 2009.

[18] Guohong Cao. Proactive Power-Aware Cache Management for Mobile Computing Systems. *IEEE Transactions on Computers*, 51(6):608–621, 2002.

[19] Jiannong Cao, Yang Zhang, Guohong Cao, and Li Xie. Data Consistency for Cooperative Caching in Mobile Environments. *IEEE Computer*, 40(4):60–66, 2007.

[20] Yuxin Cao and Tamer Özsu. Evaluation of Strong Consistency Web Caching Techniques. *World Wide Web: Internet and Web Information Systems*, 5:95–123, 2002.

[21] Jorge Cardoso, Konrad Voigt, and Matthias Winkler. Service Engineering for the Internet of Services. In *Enterprise Information Systems*, volume 19, pages 15–27. Springer, 2008.

[22] Jorge Cardoso, Alistair Barros, Norman May, and Uwe Kylau. Towards a Unified Service Description Language for the Internet of Services: Requirements and First Developments. In *IEEE International Conference on Services Computing (SCC '10)*, pages 602–609. IEEE, 2010.

[23] Hei Chan and Adnan Darwiche. On the Revision of Probabilistic Beliefs Using Uncertain Evidence. *Artificial Intelligence*, 163:67–90, 2005.

[24] Hua Yun Chen, Hui Xie, and Yi Qian. Multiple Imputation for Missing Values through Conditional Semiparametric Odds Ratio Models. *Biometrics*, 67(3): 799–809, 2011.

[25] Betty Cheng, Rogério de Lemos, Holger Giese, Paola Inverardi, Jeff Magee, Jesper Andersson, Basil Becker, Nelly Bencomo, Yuriy Brun, Bojan Cukic, Giovanna Di Marzo Serugendo, Schahram Dustdar, Anthony Finkelstein, Cristina Gacek, Kurt Geihs, Vincenzo Grassi, Gabor Karsai, Holger Kienle, Jeff Kramer, Marin Litoiu, Sam Malek, Raffaela Mirandola, Hausi Müller, Sooyong Park, Mary Shaw, Matthias Tichy, Massimo Tivoli, Danny Weyns, and Jon Whittle. Software Engineering for Self-Adaptive Systems: A Research Roadmap. In *Software Engineering for Self-Adaptive Systems*, pages 1–26. Springer, 2009.

[26] Dan Davis and Manish Parashar. Latency Performance of SOAP Implementations. In *IEEE/ACM International Symposium on Cluster Computing and the Grid (CCGRID '02)*, pages 407–412. IEEE, 2002.

[27] Konstantinos Demestichas, Artemis Koutsorodi, Evgenia Adamopoulou, and Michael Theologou. Modelling User preferences and Configuring Services in B3G Devices. *Wireless Networks*, 14(5):699–713, 2008.

[28] Simon Dobson, Spyros Denazis, Antonio Fernández, Dominique Gaïti, Erol Gelenbe, Fabio Massacci, Paddy Nixon, Fabrice Saffre, Nikita Schmidt, and Franco Zambonelli. A Survey of Autonomic Communications. *ACM Transactions on Autonomous and Adaptive Systems*, 1:223–259, 2006.

[29] Simon Dobson, Roy Sterritt, Paddy Nixon, and Mike Hinchey. Fulfilling the Vision of Autonomic Computing. *Computer*, 43(1):35–41, 2010.

[30] Schahram Dustdar and Lukasz Juszczyk. Dynamic Replication and Synchronization of Web Services for High Availability in Mobile Ad-hoc Networks. *Service Oriented Computing and Applications*, 1(1):19–33, 2007.

[31] Julian Eckert. *Cross-organizational Service-based Workflows – Solution Strategies for Quality of Service Optimization*. PhD thesis, Technische Universität Darmstadt, 2009. URL `http://tubiblio.ulb.tu-darmstadt.de/39395/`. Last accessed in January 2012.

[32] Julian Eckert, Deniz Ertogrul, Apostolos Papageorgiou, Nicolas Repp, and Ralf Steinmetz. The Impact of Service Pricing Models on Service Selection. In *International Conference on Internet and Web Applications and Services (ICIW '09)*, pages 316–321. IARIA, 2009.

[33] Julian Eckert, Tim Lehrig, Apostolos Papageorgiou, Nicolas Repp, and Ralf Steinmetz. Solving Resource Planning Problems – A Heuristical Solution. In *International Conference on Information Integration and Web-based Application and Services (iiWAS 2009)*, pages 348–353. OCG / ACM, 2009.

[34] Kamal Elbashir and Ralph Deters. Transparent Caching for Nomadic WS Clients. In *IEEE International Conference on Web Services (ICWS '05)*, pages 177–184. IEEE, 2005.

[35] Thomas Erl. *Service-oriented Architecture (SOA): Concepts, Technology, and Design*. Prentice Hall, 2005. URL `http://www.soabooks.com`. Last accessed in January 2012.

[36] Jianchun Fan and Subbarao Kambhampati. A Snapshot of Public Web Services. *ACM SIGMOD Records*, 34:24–32, 2005.

[37] Gábor Fodor, Anders Furuskär, and Johan Lundsjö. On Access Selection Techniques in Always Best Connected Networks. In *ITC Specialist Seminar on Performance Evaluation of Wireless and Mobile Systems*, 2004.

[38] Marios Fokaefs, Rimon Mikhaiel, Nikolaos Tsantalis, Eleni Stroulia, and Alex Lau. An Empirical Study on Web Service Evolution. In *IEEE International Conference on Web Services (ICWS '11)*, pages 49–56. IEEE, 2011.

[39] George Forman and John Zahorjan. The Challenges of Mobile Computing. *Computer*, 27(4):38–47, 1994.

[40] Coleen Frye. SOA Growth and Change, 2009. URL `http://searchsoa.techtarget.com/news/1351532/SOA-growth-and-change-TechTarget-survey-shows-SaaS-BPM-emerging`. Last accessed in January 2012.

[41] Vangelis Gazis, Nancy Alonistioti, and Lazaros Merakos. Toward a Generic Always best Connected Capability in Integrated WLAN/UMTS Cellular Mobile Networks. *IEEE Wireless Communications*, 12(3):20–29, 2005.

[42] Guido Gehlen and Ralf Bergs. Performance of Mobile Web Service Access using the Wireless Application Protocol (WAP). In *World Wireless Congress (WWC '04)*, pages 427–432, 2004.

[43] Brian Goodman. Accelerate your Web Services with Caching. *IBM developerWorks Journal*, 2002. URL `http://www.ibm.com/developerworks/webservices/library/ws-cach1/`. Last accessed in January 2012.

[44] Carsten Griwodz. *Wide-area True Video-on-Demand by a Decentralized Cache-based Distribution Infrastructure*. PhD thesis, Technische Universität Darmstadt, November 2000. URL `http://tuprints.ulb.tu-darmstadt.de/81/`. Last accessed in January 2012.

[45] Christin Groba and Siobhan Clarke. Web Services on Embedded Systems – A Performance Study. In *Workshop Proceedings of the IEEE International Conference on Pervasive Computing and Communications (PERCOM Workshops 2010)*, pages 726–731. IEEE, 2010.

[46] Eva Gustafsson and Annika Jonnson. Always Best Connected. *IEEE Wireless Communications*, 10(1):49–55, 2003.

[47] Mihail Halatchev and Le Gruenwald. Estimating Missing Values in Related Sensor Data Streams. In *International Conference on Management of Data*, pages 83–94. Computer Society of India, 2005.

[48] Stephen Hall and Gerald Kotonya. An Adaptable Fault-Tolerance for SOA using a Peer-to-Peer Framework. In *International Workshop on Service-Oriented System Engineering (SOSE 2007) at the IEEE International Conference on e-Business Engineering (ICEBE 2007)*, pages 520–527. IEEE, 2007.

[49] Pual Henry and Hui Luo. WiFi: What's Next? *IEEE Communications Magazine*, 40(12):66–72, 2002.

[50] Bjorn Hestnes, Peter Brooks, and Svein Heiestad. Measuring QoE for Improving the Usage of Telecommunication Services, 2009. Telenor RI Research Report R 21/2009.

[51] Frederick Hillier and Gerald Lieberman. *MP: Introduction to Operations Research*. McGraw-Hill Companies, 2004.

[52] Raj Jain. *The Art of Computer Systems Performance Analysis: Techniques for Experimental Design, Measurement, Simulation, and Modeling*. John Wiley & Sons, 1991.

[53] Predrag Jelenković and Ana Radovanović. The Persistent-Access-Caching Algorithm. *Random Structures and Algorithms*, 33(2):219–251, 2008.

[54] Chuanyi Ji and Anwar Elwalid. Measurement-based Network Monitoring: Missing Data Formulation and Scalability Analysis. In *IEEE International Symposium on Information Theory*, page 78. IEEE, 2000.

[55] Nan Jiang. A Data Imputation Model in Sensor Databases. In *International Conference on High Performance Computing and Communications (HPCC)*, pages 86–96, 2007.

[56] Hai Jin, Shadi Ibrahim, Tim Bell, Li Qi, Haijun Cao, Song Wu, and Xuanhua Shi. *Tools and Technologies for Building Clouds*, pages 3–20. Springer, 2010.

[57] Nicolai Josuttis. *SOA in Practice: The Art of Distributed System Design*. O'Reilly Media, 2007. URL `http://www.soa-in-practice.com`. Last accessed in January 2012.

[58] Achilles Kameas and Ioannis Calemis. Pervasive Systems in Health Care. In *Handbook of Ambient Intelligence and Smart Environments*, pages 315–346. Springer, 2010.

[59] Jaakko Kangasharju, Sasu Tarkoma, and Kimmo Raatikainen. Comparing SOAP Performance for Various Encodings, Protocols, and Connections. In *Personal Wireless Communications*, pages 397–406. Springer, 2003.

[60] Ramakrishna Karedla, Spencer Love, and Bradley Wherry. Caching Strategies to Improve Disk System Performance. *IEEE Computer*, 27(3):38–46, 1994.

[61] Sebastian Kaune, Konstantin Pussep, Christof Leng, Aleksandra Kovacevic, Gareth Tyson, and Ralf Steinmetz. Modelling the Internet Delay Space Based on Geographical Locations. In *Euromicro International Conference on Parallel, Distributed, and Network-based Processing (PDP 2009)*, pages 301–310. IEEE, 2009.

[62] Jeffrey Kephart and David Chess. The Vision of Autonomic Computing. *Computer*, 36(1):41–50, 2003.

[63] Ralph Kling. Intel Motes: Advanced Sensor Network Platforms and Applications. In *IEEE MTT-S International Microwave Symposium Digest (MWSYM '05)*, pages 365–368. IEEE, 2005.

[64] Andreas Koehler and Claudia Som. Effects of Pervasive Computing on Sustainable Development. *Technology and Society Magazine*, 24(1):15–23, 2005.

[65] Gerald Kotonya and Stephen Hall. A Differentiation-Aware Fault-Tolerant Framework for Web Services. In *International Conference on Service-Oriented Computing (ICSOC 2010)*, pages 137–151. Springer, 2010.

[66] Dirk Krafzig, Karl Banke, and Dirk Slama. *Enterprise SOA: Service-oriented Architecture Best Practices*. Prentice Hall, 2004.

[67] Dirk Krafzig, Karl Banke, and Dirk Slama. *Enterprise SOA*. Pearson Education, 2005.

[68] Axel Küpper, Georg Treu, and Claudia Linnhoff-Popien. TraX: A Device-centric Middleware Framework for Location-based Services. *IEEE Communications Magazine*, 44(9):114–120, 2006.

[69] Alexandros Labrinidis and Nick Roussopoulos. Exploring the Tradeoff between Performance and Data Freshness in Database-Driven Web servers. *The VLDB Journal*, 13:240–255, 2004.

[70] Kwong Yuen Lai, Thi Khoi Anh Phan, and Zahir Tari. Efficient SOAP Binding for Mobile Web Services. In *IEEE Conference on Local Computer Networks (LCN '05)*, pages 218–225. IEEE, 2005.

[71] Ulrich Lampe. Optimizing the Distribution of Software Services in Infrastructure Clouds. In *IEEE World Congress on Services (SERVICES 2011)*, pages 69–72. IEEE, 2011.

[72] Frank Leymann. The (Service) Bus: Services Penetrate Everyday Life. In *International Conference on Service Oriented Computing (ICSOC '05)*, pages 12–20. Springer, 2005.

[73] Wei Li, Yang Xiao, and Yi Pan. *Adaptation Techniques in Wireless Multimedia Networks*. Nova Science Publishers, Inc., 2006.

[74] Wubin Li, Zhuofeng Zhao, Kaiyuan Qi, Jun Fang, and Weilong Ding. A Consistency-Preserving Mechanism for Web Services Response Caching. In *IEEE International Conference on Web Services (ICWS '08)*, pages 683–690. IEEE, 2008.

[75] Yan Li, Yao Liu, Liangjie Zhang, Ge Li, Bing Xie, and Jiasu Sun. An Exploratory Study of Web Services on the Internet. In *IEEE International Conference on Web Services (ICWS '07)*, pages 380–387. IEEE, 2007.

[76] YuanYuan Li and Lynne Parker. A Spatial-Temporal Imputation Technique for Classification with Missing Data in a Wireless Sensor Network. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2008)*, pages 3272–3279. IEEE, 2008.

[77] Claudia Linnhoff-Popien and Heinz-Gerd Hegering. *Trends in Distributed Systems: Towards a Universal Service Market*. Springer, 2000.

[78] Claudia Linnhoff-Popien and Stephan Verclas. *Mit Business-Apps ins Zeitalter mobiler Geschäftsprozesse*, pages 3–16. Springer, 2012.

[79] Xin Liu and Ralph Deters. An Efficient Dual Caching Strategy for Web Service-Enabled PDAs. In *ACM Symposium on Applied Computing (SAC '07)*, pages 788–794. ACM, 2007.

[80] Jörg Lonthoff and Erich Ortner. Klassifikations- und Lösungsansätze für Web Services im Mobilen Umfeld. Technical report, Darmstadt Technical University, Department of Business Administration, Economics and Law, Institute for Business Studies (BWL), 2006.

[81] David Luckham. *The Power of Events: An Introduction to Complex Event Processing in Distributed Enterprise Systems*. Addison-Wesley Longman Publishing Co., 2001.

[82] Nazir Malik, Umar Mahmud, and Younus Javed. Future Challenges in Context-Aware Computing. In *IADIS International Conference WWW/Internet*, pages 306–310. IADIS Press, 2007.

[83] Daniel Menascé. QoS Issues in Web Services. *IEEE Internet Computing*, 6(6): 72–75, 2002.

[84] André Miede. *Cross-organizational Service Security – Attack Modeling and Evaluation of Selected Countermeasures*. Dr. Hut Verlag, 2010.

[85] André Miede, Steffen Braun, Julian Eckert, Dieter Schuller, Nicolas Repp, and Ralf Steinmetz. A Comparison of Self-Organization Mechanisms in Nature and Information Technology. In *Americas Conference on Information Systems (AMCIS '09)*. Association for Information Systems, 2009.

[86] André Miede, Ulrich Lampe, Dieter Schuller, Julian Eckert, and Ralf Steinmetz. Evaluating the QoS Impact of Web Service Anonymity. In *IEEE European Conference on Web Services (ECOWS 2010)*, pages 75–82. IEEE, 2010.

[87] Sameena Salvucci Ming-Xiu Hu. A Study of Imputation Algorithms, September 2001. URL `http://nces.ed.gov/pubs2001/200117.pdf`. Working Paper 200117, National Center for Education Statistics, U.S. Department of Education.

[88] Parag Mogre, Nico d'Heureuse, Matthias Hollick, and Ralf Steinmetz. CORE: Centrally Optimized Routing Extensions for the IEEE 802.16 MeSH Mode. In *IEEE Conference on Local Computer Networks (LCN 2008)*, pages 58–65. IEEE, 2008.

[89] Paul Müller and Bernd Reuther. Future Internet Architecture – A Service-oriented Approach. *it – Information Technology*, 50(6):383–389, 2008.

[90] Ingunn Myrtveit, Erik Stensrud, and Ulf Olsson. Analyzing Data Sets with Missing Data: An Empirical Evaluation of Imputation Methods and Likelihood-Based Methods. *IEEE Transactions on Software Engineering*, 27(11):999–1013, 2001.

[91] Klara Nahrstedt, Dongyan Xu, Duangdao Wichadakul, and Baochun Li. QoS-aware Middleware for Ubiquitous and Heterogeneous Environments. *IEEE Communications Magazine*, 39(11):140–148, 2001.

[92] Klara Nahrstedt, Ahsan Arefin, Raoul Rivas, Pooja Agarwal, Zixia Huang, Wanmin Wu, and Zhenyu Yang. QoS and Resource Management in Distributed Interactive Multimedia Environments. *Multimedia Tools and Applications*, 51: 99–132, 2011.

[93] Michael Niemann, Julian Eckert, Nicolas Repp, and Ralf Steinmetz. Towards a Generic Governance Model for Service-oriented Architectures. In *Americas Conference on Information Systems (AMCIS '08)*. Association for Information Systems, AIS, 2008.

[94] Paddy Nixon and Vinny Cahill. Mobile Computing: Technologies for a Disconnected Society. *Internet Computing*, 2(1):19–21, 1998.

[95] Daniel Oberle, Nadeem Bhatti, Saartje Brockmans, Michael Niemann, and Christian Janiesch. Countering Service Information Challenges in the Internet of Services. *Business and Information Systems Engineering*, 1(5):370–390, 2009.

[96] ITU-T (Telecommunication Standardization Sector of the International Telecommunication Unit). Definition of Quality of Experience, 2007. Liaison Statement, ref. TD 109rev2 (PLEN/12).

[97] Sangyoon Oh and Geoffrey Fox. Optimizing Web Service Messaging Performance in Mobile Computing. *Future Generation Computer Systems*, 23:623–632, 2007.

[98] Guadalupe Ortiz and Alfonso Garcia de Prado. Towards Adapting Web Services for Multiple Devices. In *International Conference on Internet and Web Applications and Services (ICIW '09)*, pages 292–297. IARIA, 2009.

[99] Apostolos Papageorgiou, Bastian Leferink, Julian Eckert, Nicolas Repp, and Ralf Steinmetz. Bridging the Gaps towards Structured Mobile SOA. In *International Conference on Advances in Mobile Computing & Multimedia (MoMM '09)*, pages 288–294. OCG/ACM, 2009.

[100] Apostolos Papageorgiou, Jeremias Blendin, André Miede, Julian Eckert, and Ralf Steinmetz. Study and Comparison of Adaptation Mechanisms for Performance Enhancements of Mobile Web Service Consumption. In *IEEE World Congress on Services (SERVICES 2010)*, pages 667–670. IEEE, 2010.

[101] Apostolos Papageorgiou, Tronje Krop, Sebastian Ahlfeld, Stefan Schulte, Julian Eckert, and Ralf Steinmetz. Enhancing Availability with Self-Organization Extensions in a SOA Platform. In *International Conference on Internet and Web Applications and Services (ICIW 2010)*, pages 161–166. IARIA, 2010.

[102] Apostolos Papageorgiou, Tronje Krop, Sebastian Ahlfeld, Stefan Schulte, Julian Eckert, and Ralf Steinmetz. Enhancing Availability through Dynamic Monitoring and Management in a Self-Adaptive SOA Platform. *International Journal on Advances in Software*, 3(3&4):434–446, Feb 2011.

[103] Apostolos Papageorgiou, André Miede, Dieter Schuller, Stefan Schulte, and Ralf Steinmetz. Always Best Served: On the behaviour of QoS- and QoE-based Algorithms for Web Service Adaptation. In *Workshop Proceedings of the IEEE*

*International Conference on Pervasive Computing and Communications (PERCOM Workshops 2011)*, pages 71–76. IEEE, 2011.

[104] Apostolos Papageorgiou, Marius Schatke, Stefan Schulte, and Ralf Steinmetz. Enhancing the Caching of Web Service Responses on Wireless Clients. In *IEEE International Conference on Web Services (ICWS '11)*, pages 9–16, 2011.

[105] Michael Papazoglou and Willem-Jan Heuvel. Service Oriented Architectures: Approaches, Technologies and Research Issues. *The VLDB Journal*, 16(3):389–415, 2007.

[106] Michael Papazoglou, Paolo Traverso, Schahram Dustdar, and Frank Leymann. Service-Oriented Computing: State of the Art and Research Challenges. *IEEE Computer*, 40(11):38–45, 2007.

[107] Cisco White Paper. Cisco Visual Networking Index - Global Mobile Data Traffic Forecast, 2011. URL `http://www.cisco.com/en/US/solutions/collateral/ns341/ns525/ns537/ns705/ns827/white_paper_c11-520862.html`. Last accessed in January 2012.

[108] IBM White Paper. An Architectural Blueprint for Autonomic Computing. URL `http://www-01.ibm.com/software/tivoli/autonomic/pdfs/AC_Blueprint_White_Paper_4th.pdf`. Last accessed in January 2012.

[109] Motorola White Paper. Realistic LTE Performance – From Peak Rate to Subscriber Experience, 2009. URL `http://www.motorola.com/web/Business/_Documents/static%20files/Realistic_LTE_Experience_White_Paper_FINAL.pdf`. Last accessed in January 2012.

[110] Cesare Pautasso, Olaf Zimmermann, and Frank Leymann. Restful Web Services vs. "big" Web Services: Making the Right Architectural Decision. In *International Conference on World Wide Web (WWW '08)*, pages 805–814. ACM, 2008.

[111] Judea Pearl. *Causality: Models, Reasoning, and Inference*. Cambridge University Press, New York, USA, 2000.

[112] Kostas Pentikousis. In Search of Energy-efficient Mobile Networking. *IEEE Communications Magazine*, 48(1):95–103, 2010.

[113] Gian Paolo Perrucci, Frank Fitzek, Giovanni Sasso, Wolfgang Kellerer, and Jörg Widmer. On the Impact of 2G and 3G Network Usage for Mobile Phones' Battery Life. In *European Wireless Conference (EW 2009)*, pages 255–259. IEEE, 2009.

[114] Mikko Perttunen, Jukka Riekki, and Ora Lassila. Context Representation and Reasoning in Pervasive Computing: A Review. *International Journal of Multimedia and Ubiquitous Engineering*, 4:1–28, 2009.

[115] Christy Pettey and Holly Stevens. Gartner Identifies the Top 10 Consumer Mobile Applications for 2012, 2009. URL `http://www.gartner.com/it/page.jsp?id=1230413`. Last accessed in January 2012.

[116] Khoi Anh Phan, Zahir Tari, and Peter Bertok. A Benchmark on SOAP's Transport Protocols Performance for Mobile Applications. In *ACM Symposium on Applied Computing (SAC '06)*, pages 1139–1144. ACM, 2006.

[117] Stefan Podlipnig and Laszlo Böszörmenyi. A Survey of Web Cache Replacement Strategies. *ACM Computing Surveys*, 35(4):331–373, 2003.

[118] Larry Press. Personal Computing: The Post-PC Era. *Communications of the ACM*, 42:21–24, 1999.

[119] Ahmad Rahmati and Lin Zhong. Context-for-Wireless: Context-Sensitive Energy-efficient Wireless Data Transfer. In *International Conference on Mobile Systems, Applications and Services (MobiSys '07)*, pages 165–178. ACM, 2007.

[120] Geoffrey Raines. Cloud Computing and SOA. Technical Report 09-0743, The MITRE Corporation, 2009. URL `http://www.mitre.org/work/tech_papers/tech_papers_09/09_0743/`. Last accessed in January 2012.

[121] Andreas Reinhardt, Parag Mogre, and Ralf Steinmetz. Lightweight Remote Procedure Calls for Wireless Sensor and Actuator Networks. In *Workshop Proceedings of the IEEE International Conference on Pervasive Computing and Communications (PERCOM Workshops 2011)*, pages 116–121. IEEE, 2011.

[122] Nicolas Repp. *Überwachung und Steuerung dienstbasierter Architekturen – Verteilungsstrategien und deren Umsetzung*. Books on Demand, 2009.

[123] Nicolas Repp, André Miede, Michael Niemann, and Ralf Steinmetz. WS-Re2Policy: A Policy Language for Distributed SLA Monitoring and Enforcement. In *International Conference on Systems and Networks Communications (ICSNC '08)*, pages 256–261. IEEE, 2008.

[124] Ivica Rimac. *End-to-End Mechanisms for Rate-Adaptive Multicast Streaming over the Internet*. PhD thesis, Technische Universität Darmstadt, 2005. URL `http://tuprints.ulb.tu-darmstadt.de/520/`.

[125] Jörg Roth. *Mobile Computing*. dpunkt Verlag, 2005.

[126] Donald Rubin. Inference and Missing Data. *Biometrika*, 63(3):581–592, 1976.

[127] Donald Rubin. *Multiple Imputation for Nonresponse in Surveys*. John Wiley & Sons, 1987.

[128] Xianyi Rui. Missing Data Imputation Based on Compressive Sensing for Robust Speaker Identification. In *International Conference on Wireless Communications and Signal Processing (WCSP '10)*, pages 1–4. IEEE, 2010.

[129] Mazeiar Salehie and Ladan Tahvildari. Self-Adaptive Software: Landscape and Research Challenges. *ACM Transactions on Autonomous and Adaptive Systems*, 4: 14:1–14:42, 2009.

[130] Paul Sandoz, Santiago Pericas-Geertsen, Kohuske Kawaguchi, Marc Hadley, and Eduardo Pelegri-Llopart. Fast Web Services, 2003. URL `http://java.sun.com/developer/technicalArticles/WebServices/fastWS/`. Last accessed in January 2012.

[131] Joseph Schafer and John Graham. Missing data: Our view of the State of the Art. *Psychological Methods*, 7(2):147–177, 2002.

[132] Alexander Schill. Remote Procedure Call: Fortgeschrittene Konzepte und Systeme – ein Überblick, Teil 1: Grundlagen. *Informatik-Spektrum*, 15(2):79–87, 1992.

[133] Jens Schmitt. *Heterogeneous Network Quality of Service Systems*. Kluwer Academic Publishers, 2001.

[134] Johannes Schmitt. *Anpassungsfähige Kontextbestimmung zur Unterstützung von Kommunikationsdiensten*. PhD thesis, Technische Universität Darmstadt, 2009. URL `http://tuprints.ulb.tu-darmstadt.de/1997/`.

[135] Daniel Schreiber, Erwin Aitenbichler, Andreas Göb, and Max Mühlhäuser. Reducing User Perceived Latency in Mobile Processes. In *IEEE International Conference on Web Services (ICWS '10)*, pages 235–242. IEEE, 2010.

[136] Christoph Schroth. The Internet of Services: Global Industrialization of Information Intensive Services. In *IEEE International Conference on Digital Information Management (ICDIM 2007)*, pages 635–642. IEEE, 2007.

[137] Dieter Schuller, André Miede, Julian Eckert, Ulrich Lampe, Apostolos Papageorgiou, and Ralf Steinmetz. QoS–based Optimization of Service Compositions for Complex Workflows. In *International Conference on Service Oriented Computing (ICSOC 2010)*, pages 641–648. Springer, 2010.

[138] Dieter Schuller, Artem Polyvyanyy, Luciano Garcia-Banuelos, and Stefan Schulte. Optimization of Complex QoS-aware Service Compositions. In *International Conference on Service Oriented Computing (ICSOC 2011)*, pages 452–466. Springer, 2011.

[139] Stefan Schulte. *Web Service Discovery Based on Semantic Information – Query Formulation and Adaptive Matchmaking*. PhD thesis, Technische Universität Darmstadt, 2010. URL `http://tuprints.ulb.tu-darmstadt.de/2293/`. Last accessed in January 2012.

[140] Stefan Schulte, Melanie Siebenhaar, Julian Eckert, and Ralf Steinmetz. Query Languages for Semantic Web Services. 176(2):109–114, 2010.

[141] Amit Sehgal and Rajeev Agrawal. QoS-based Network Selection Scheme for 4G Systems. *IEEE Transactions on Consumer Electronics*, 56(2):560–565, 2010.

[142] Stefania Sesia, Issam Toufik, and Matthow Baker. *LTE, The UMTS Long Term Evolution: From Theory to Practice*. John Wiley & Sons, UK, 2009.

[143] Jinyang Shi, Yuehui Jin, Hui Huang, and Dajiang Zhang. Experimental Performance Studies of SCTP in Wireless Access Networks. In *International Conference on Communication Technology (ICCT 2003)*, pages 392–395. IEEE, 2003.

[144] Alex Shye, Benjamin Scholbrock, and Gokhan Memik. Into the Wild: Studying Real User Activity Patterns to Guide Power Optimizations for Mobile Architectures. In *Annual IEEE/ACM International Symposium on Microarchitecture (MICRO 42)*, pages 168–178. ACM, 2009.

[145] Melanie Siebenhaar, Ulrich Lampe, Tim Lehrig, Sebastian Zöller, Stefan Schulte, and Ralf Steinmetz. Complex Service Provisioning in Collaborative Cloud Markets. In *European Conference ServiceWave*, pages 88–99. Springer, 2011.

[146] Alan Smith. Disk Cache – Miss Ratio Analysis and Design Considerations. *ACM Transactions on Computer Systems*, 3, 1985.

[147] Timo Smura, Antero Kivi, and Juuso Toyli. A Framework for Analysing the Usage of Mobile Services. *The Journal of Policy, Regulation and Strategy for Telecommunications*, 11(4):53–67, 2009.

[148] David Soldani, Man Li, and Renaud Cuny. *QoS and QoE Management in UMTS Cellular Systems*. John Wiley & Sons, UK, 2006.

[149] Ralf Steinmetz and Klara Nahrstedt. *Multimedia Systems*. Springer, 2004.

[150] Ralf Steinmetz and Klaus Wehrle. *Peer-to-Peer Systems and Applications*. Springer, 2005.

[151] Heinz Stockinger. Defining the Grid: A Snapshot on the Current View. *The Journal of Supercomputing*, 42:3–17, 2007.

[152] Toshiro Takase and Michiaki Tatsubori. Efficient Web Services Response Caching by Selecting Optimal Data Representation. In *International Conference on Distributed Computing Systems (ICDCS '04)*, pages 188–197. IEEE, 2004.

[153] Xueyan Tang, Jianliang Xu, and Wang-Chien Lee. Analysis of TTL-Based Consistency in Unstructured Peer-to-Peer Networks. *IEEE Transactions on Parallel and Distributed Systems*, 19(12):1683–1694, 2008.

[154] Junichi Tatemura, Oliver Po, Arsany Sawires, Divyakant Agrawal, and Selçuk Candan. WReX: A Scalable Middleware Architecture to Enable XML Caching for Web-Services. In *ACM/IFIP/USENIX 2005 International Conference on Middleware (Middleware '05)*, pages 124–143. Springer, 2005.

[155] Beng Hang Tay and Akkihebbal Ananda. A Survey of Remote Procedure Calls. *SIGOPS Operating Systems Review*, 24:68–79, 1990.

[156] Joe Tekli, Ernesto Damiani, and Richard Chbeir. Differential SOAP Multicasting. In *IEEE International Conference on Web Services (ICWS '11)*, pages 1–8. IEEE, 2011.

[157] Joe Tekli, Ernesto Damiani, Richard Chbeir, and Gabriele Gianini. SOAP Processing Performance and Enhancement. *IEEE Transactions on Services Computing*, PP(99):1–18, 2011.

[158] Renne Tergujeff, Jyrki Haajanen, Juha Leppanen, and Santtu Toivonen. Mobile SOA: Service Orientation on Lightweight Mobile Devices. In *IEEE International Conference on Web Services (ICWS '07)*, pages 1224–1225. IEEE, 2007.

[159] Douglas Terry and Venugopalan Ramasubramanian. Caching XML Web Services for Mobility. *ACM Queue*, 1(3):70–78, 2003.

[160] Min Tian, Thiemo Voigt, Tomasz Naumowicz, Hartmut Ritter, and Jochen Schiller. Performance Considerations for Mobile Web Services. *Computer Communications*, 27:1097–1105, 2004.

[161] Davide Tosi, Giovanni Denaro, and Mauro Pezze. Towards Autonomic Service-Oriented Applications. *International Journal of Autonomic Computing*, 1(1):58–80, 2009.

[162] Stef Van Buuren and Karin Groothuis-Oudshoorn. MICE: Multivariate Imputation by Chained Equations in R. *Journal of Statistical Software*, 45(3):1–67, 2011.

[163] Matthew Walker, Rory Turnbull, and Nicholas Sim. Future Mobile Devices – An Overview of Emerging Device Trends, and the Impact on Future Converged Services. *BT Technology Journal*, 25:120–125, 2007.

[164] Guijun Wang, Changzhou Wang, Alice Chen, Haiqin Wang, Casey Fung, Stephen Uczekaj, Yi-Liang Chen, Wayne Guthmiller Guthmiller, and Joseph Lee. Service Level Management using QoS Monitoring, Diagnostics, and Adaptation for Networked Enterprise Systems. In *IEEE International EDOC Enterprise Computing Conference (EDOC '05)*, pages 239–250. IEEE, 2005.

[165] Ning Wang, Michael Welzl, and Liang Zhang. A High Performance SOAP Engine for Grid Computing. In *Networks for Grid Applications (Gridnets 2008)*, volume 2, pages 1–8. Springer, 2009.

[166] Sanjiva Weerawarana, Francisco Curbera, Frank Leymann, Tony Storey, and Donald Ferguson. *Web Services Platform Architecture: SOAP, WSDL, WS-Policy, WS-Addressing, WS-BPEL, WS-Reliable Messaging and More*. Prentice Hall, 2005.

[167] Yi Wei and Brian Blake. Service-Oriented Computing and Cloud Computing: Challenges and Opportunities. *Internet Computing*, 14(6):72–75, 2010.

[168] Christian Werner, Carsten Buschmann, and Stefan Fischer. WSDL-Driven SOAP Compression. *International Journal of Web Services Research*, 2(1):18–35, 2005.

[169] Patrick Widener, Greg Eisenhauer, Karsten Schwan, and Fabian Bustamante. Open Metadata Formats: Efficient XML-Based Communication for High Performance Computing. *Cluster Computing*, 5:315–324, 2002.

[170] Ian Witten and Eibe Frank. *Data Mining: Practical Machine Learning Tools and Techniques*. Morgan Kaufmann Publishers, 2005.

[171] Andreas Wombacher. A-posteriori Detection of Sensor Infrastructure Errors in Correlated Sensor Data and Business Workflows. In *International Conference on Business Process Management (BPM '11)*, pages 329–344. Springer, 2011.

[172] Andreas Wombacher and Chen Li. Alternative Approaches for Workflow Similarity. In *IEEE International Conference on Services Computing (SCC '10)*, pages 337–345. IEEE, 2010.

[173] Jianhua Wu, Qinbao Song, and Junyi Shen. Missing Nominal Data Imputation Using Association Rule Based on Weighted Voting Method. In *IEEE International Joint Conference on Neural Networks (IJCNN 2008)*, pages 1157–1162. IEEE, 2008.

[174] Bo Xing and Nalini Venkatasubramanian. Multi-Constraint Dynamic Access Selection in Always Best Connected Networks. In *International Conference on Mobile and Ubiquitous Systems (MobiQuitous 2005)*, pages 56–64. IEEE, 2005.

[175] Chen Yiping and Yang Yuhang. A New 4G Architecture Providing Multimode Terminals Always Best Connected Services. *IEEE Wireless Communications*, 14 (2):36–41, 2007.

[176] Sonja Zaplata, Viktor Dreiling, and Winfried Lamersdorf. Realizing Mobile Web Services for Dynamic Applications. In *IFIP Conference on e-Business, e-Services, and e-Society (I3E 2009)*, pages 240–254. Springer, 2009.

[177] Jia Zhang and Jen-Yao Chung. An Open Framework Supporting Multimedia Web Services. *Multimedia Tools and Applications*, 30:149–164, 2006.

[178] Liang-Jie Zhang, Bing Li, and Yu Song. Mobile Services Computing. In *Enabling Technologies for Wireless E-Business*, pages 299–311. Springer, 2006.

[179] Lide Zhang, Birjodh Tiwana, Zhiyun Qian, Zhaoguang Wang, Robert P. Dick, Zhuoqing Morley Mao, and Lei Yang. Accurate Online Power Estimation and Automatic Battery Behavior Based Power Model Generation for Smartphones. In *IEEE/ACM/IFIP International Conference on Hardware/Software Codesign and System Synthesis (CODES/ISSS '10)*, pages 105–114. ACM, 2010.

[180] Ming Zhong and Satish Sharma. Development of Improved Models for Imputing Missing Traffic Counts. *Open Transportation Journal*, 3:35–48, 2009.

[181] Hubert Zimmermann. OSI Reference Model – The ISO Model of Architecture for Open Systems Interconnection. *IEEE Transactions on Communications*, 28(4): 425–432, 1980.

[182] Sebastian Zöller, Andreas Reinhardt, Stefan Schulte, and Ralf Steinmetz. Scoresheet-based Event Relevance Determination for Energy Efficiency in Wireless Sensor Networks. In *IEEE Conference on Local Computer Networks (LCN 2011)*, pages 207–210. IEEE, 2011.

# LIST OF FIGURES

# LIST OF TABLES

# LIST OF ACRONYMS

| | |
|---|---|
| ABC | Always Best Connected . . . . . 93 |
| ABS | Always Best Served . . . . . 93 |
| BN | Bayesian Network . . . . . 103 |
| BPEL | Business Process Execution Language . . . . . 12 |
| BPMN | Business Process Model and Notation . . . . . 12 |
| CPU | Central Processing Unit . . . . . 14 |
| DCC | Direct Calls with Caching . . . . . 74 |
| DCN | Direct Calls, No caching . . . . . 74 |
| DNS | Domain Name System . . . . . 8 |
| DSS | Decision Support Systems . . . . . 21 |
| ebXML | Electronic Business using eXtensible Markup Language . . . . . 12 |
| EDGE | Enhanced Data Rates for GSM Evolution . . . . . 16 |
| ESB | Enterprise Service Bus . . . . . 12 |
| ESP | Event Stream Processing . . . . . 29 |
| GHz | Gigahertz . . . . . 16 |
| GPRS | General Radio Packet Service . . . . . 16 |
| HHFR | Handheld Flexible Representation . . . . . 50 |
| HSPA | High-Speed Packet Access . . . . . 16 |
| IBM | International Business Machines . . . . . 27 |
| IEEE | Institute of Electrical and Electronics Engineers . . . . . 16 |
| IoS | Internet of Services . . . . . 1 |
| IP | Internet Protocol . . . . . 8 |
| IT | Information Technology . . . . . 8 |
| ITU | International Telecommunication Union . . . . . 18 |
| kbit/s | Kilobits per second . . . . . 17 |
| LFU | Least Frequently Used . . . . . 60 |
| LRU | Least Recently Used . . . . . 60 |
| LTE | Long Term Evolution of 3G mobile networks . . . . . 16 |
| MAR | Missing At Random . . . . . 22 |
| Mbit/s | Megabits per second . . . . . 16 |
| MCAR | Missing Completely at Random . . . . . 22 |
| MML | Mobility Mediation Layer . . . . . 39 |
| MNAR | Missing Not at Random . . . . . 22 |
| OAR | Observed At Random . . . . . 22 |
| OSI | Open Systems Interconnection . . . . . 96 |
| PC | Personal Computers . . . . . 1 |

| | |
|---|---|
| PCV | Proxied Calls with Validity-checks . . . 63 |
| PDA | Personal Digital Assistants . . . 1 |
| POI | Points-Of-Interest . . . 84 |
| QoE | Quality of Experience . . . 7 |
| QoI | Quality of Information . . . 34 |
| QoS | Quality of Service . . . 7 |
| REST | Representational State Transfer . . . 12 |
| RMI | Remote Method Invocation . . . 50 |
| RPC | Remote Procedure Call . . . 12 |
| SAML | Security Assertion Markup Language . . . 12 |
| SAWSDL | Semantic Annotations for WSDL . . . 11 |
| SCA | Service Component Architecture . . . 29 |
| SCTP | Stream Control Transmission Protocol . . . 50 |
| SMTP | Simple Mail Transfer Protocol . . . 50 |
| SOA | Service-Oriented Architecture . . . 1 |
| SOAP | Simple Object Access Protocol . . . 12 |
| SOC | Service-Oriented Computing . . . 1 |
| SoU | SOAP-over-UDP . . . 48 |
| SPARQL | SPARQL Protocol and RDF Query Language . . . 12 |
| TCP | Transmission Control Protocol . . . 11 |
| TTL | Time-To-Live . . . 20 |
| UC | Use Cases . . . 84 |
| UDDI | Universal Description, Discovery and Integration . . . 12 |
| UDP | User Datagram Protocol . . . 11 |
| UMTS | Universal Mobile Telecommunications System . . . 16 |
| USDL | Unified Service Description Language . . . 8 |
| WADL | Web Application Description Language . . . 11 |
| WAN | Wide Area Networks . . . 17 |
| WAP | Wireless Application Protocol . . . 50 |
| WEF | WSDM Event Format . . . 28 |
| WLAN | Wireless Local Area Networks . . . 16 |
| WMN | Wide Mesh Networks . . . 17 |
| WPAN | Wireless Personal Area Networks . . . 16 |
| WS | Web services . . . 2 |
| WSDL | Web Service Description Language . . . 8 |
| WSDM | Web Services Distributed Management . . . 28 |
| WSPG | Web Service Proxy Generator . . . 68 |
| WWW | World Wide Web . . . 19 |

# A

# APPENDIX

»It ain't over till the fat lady sings.«

— Anonymous

This appendix provides additional information and screenshots from the operation of the Mobility Mediation Layer (A.1), as well as further details and results (A.2 and A.3) from the conducted evaluations.

## A.1 OPERATION AND USE CASES OF THE MOBILITY MEDIATION LAYER

The Mobility Mediation Layer (MML – cf. Chapter 4, Figure 9), which has framed the scenario of the adaptation mechanisms examined in this thesis, has been developed in the context of the project Green Mobility[1]. Although the MML is a general-purpose solution for the mediation of mobile service calls in the Internet of Services, its mechanisms have been tested and/or demonstrated with particular prototypical implementations.

The current Appendix section provides briefly described screenshots that can help understand the role of the MML. The presented screenshots stem from or are related to one of the following three aspects:

- *Operation and administration:* In order to control and monitor the operation of the MML, a simple Web-based interface has been developed. The navigation through its options can support the understanding of some of the MML functionalities. fig:mmladmin-login shows the list of services that are known to the MML, i.e., the services the descriptions of which have been uploaded. These services (and their corresponding proxies) can be activated and deactivated. By clicking on "Show Logs", one can view the service call logs of particular services (cf. Figure 31). These logs include information such as the sizes of the original Web service response and the MML Web service response, as well as response times etc. The Web interface can also be used in order to upload new service descriptions, as shown in Figure 32. Finally, Figure 33 shows how the caching proxy of a service can be activated or deactivated through the MML Web interface.

- *Use Cases:* The first use cases, the examination of which led to the conception and motivated the development of the MML, have been so-called "mobile companion" applications, i.e., applications that assist a worker, a working team, or any person with similar needs in a mobile environment. Accordingly, Figure 34 presents screenshots, firstly, from a "visitor's assistant" (on the left), who can locate points of interest, find routes, search for and communicate with her/his colleagues, and more, and, secondly, from an "emergency worker's

1 `http://www.greenmobility-project.de` (Last accessed in January 2012)

assistant" (on the right), who can view the location and the status of the cooperating forces, and communicate with them. However, the main use case, with which the MML has been tested during its development and operation, has been a mobile car sharing platform (cf. Figure 35). The corresponding mobile application integrates diverse services, such as services for the spontaneous search of offered rides (Figure 35 – left) or the indication of further information about arranged rides (Figure 35 – right).

- *Single Tests and Demos:* Various further mobile client applications have been developed in order to test or demonstrate particular aspects of the mediated mobile service consumption. An simple example which demonstrates the reduction of the service response sizes achieved through the activation of diverse proxies of the MML is shown in Figure 36.

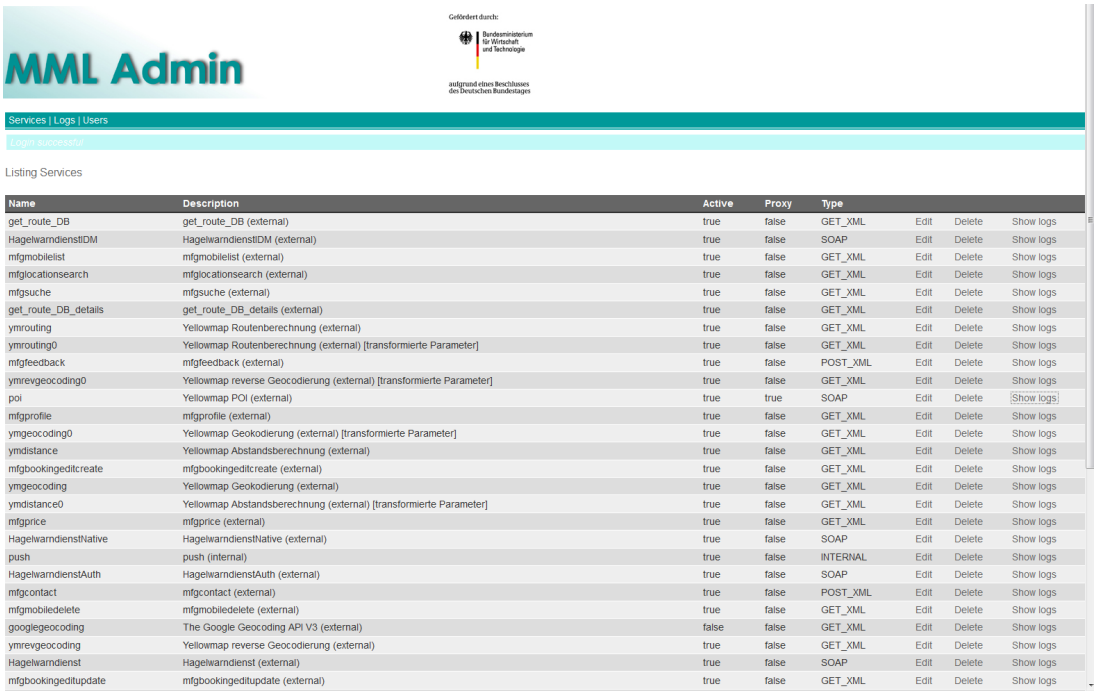



Figure 30: MML Web interface: list of services that can be mediated

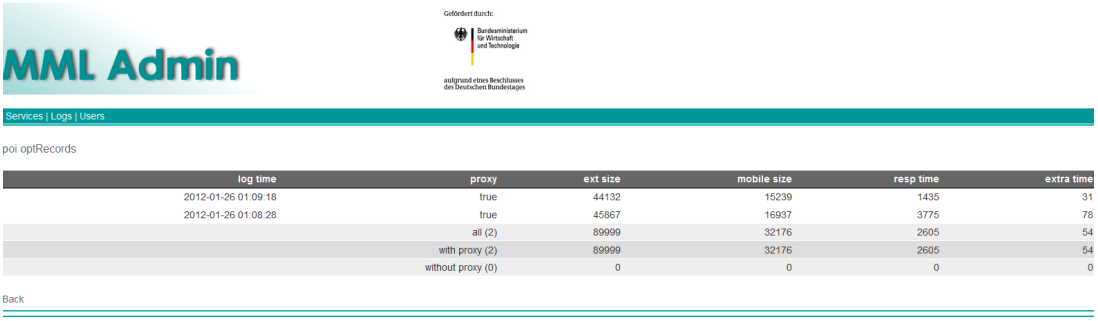



Figure 31: MML Web interface: log of mediated invocations of a particular service

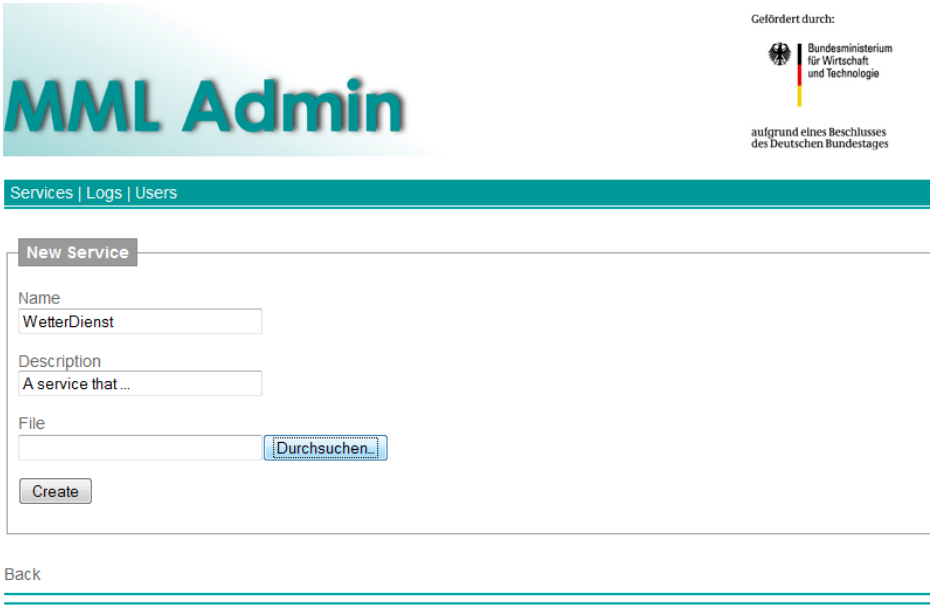



Figure 32: MML Web interface: Uploading a new service description so that invocations of this service can be mediated

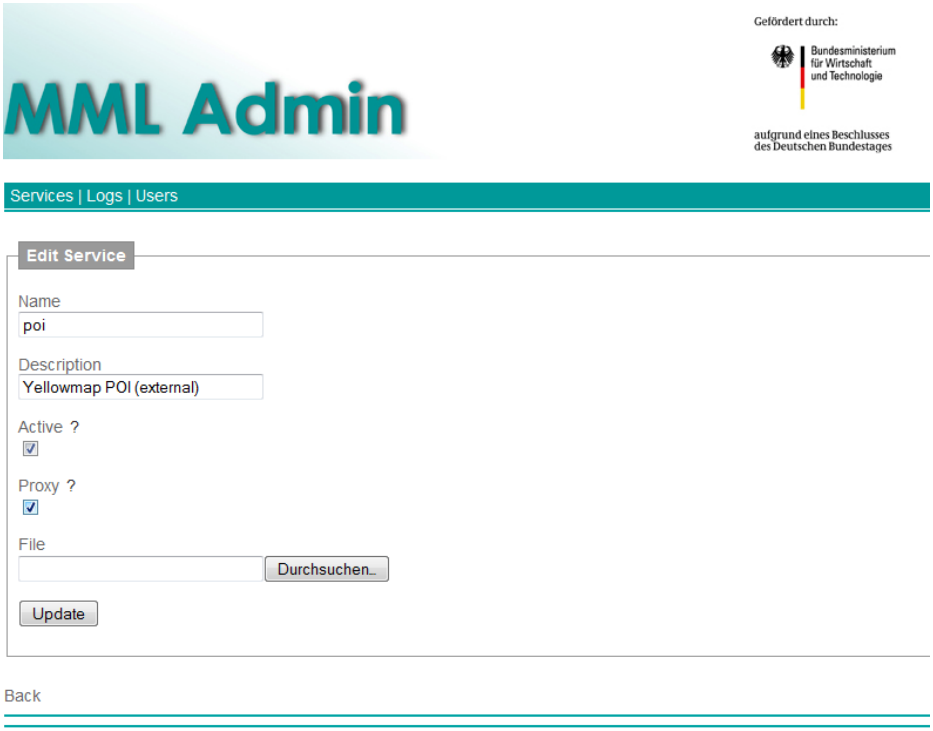



Figure 33: MML Web interface: (de-)activation of the (caching) proxy for a particular service

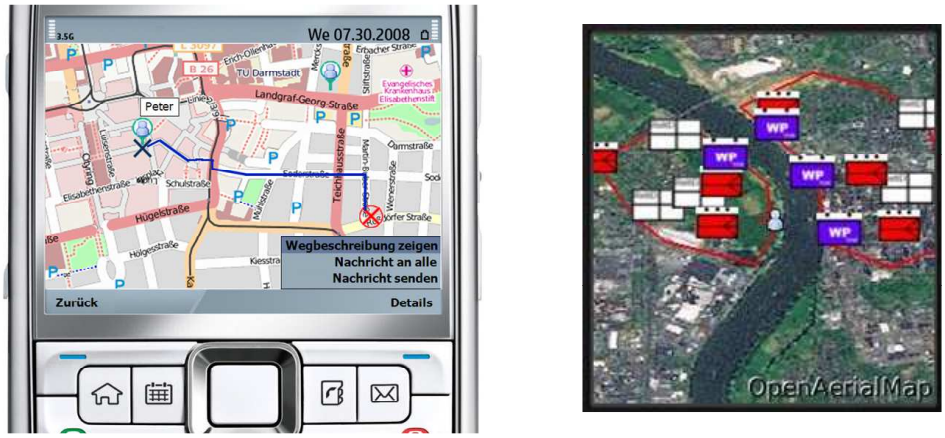



Figure 34: The mobile companion use case, which motivated the conception of the MML


Figure 35: The mobile car sharing use case – main MML use case

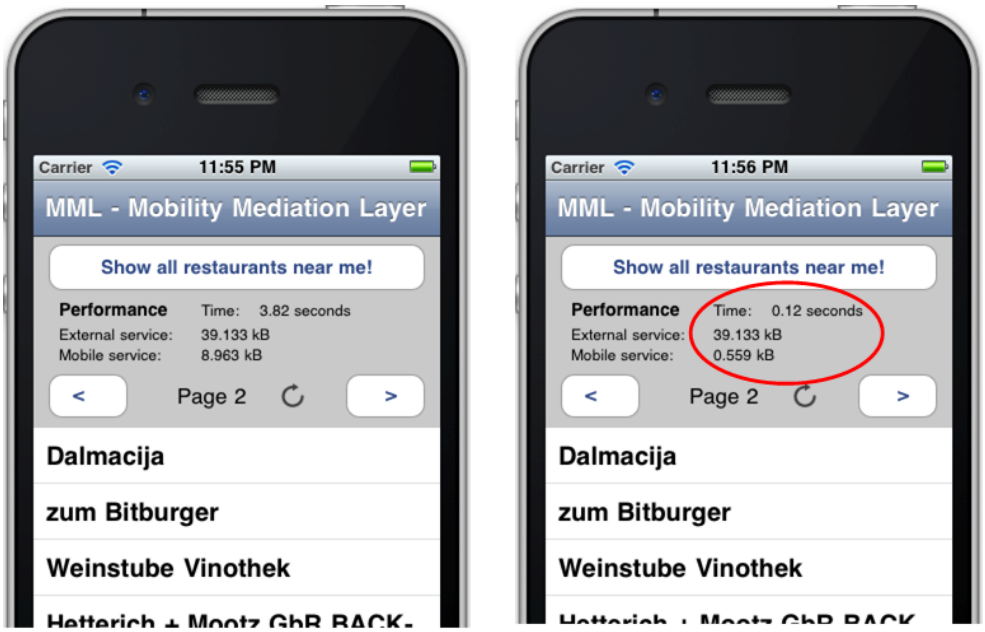



Figure 36: Simple mobile application for testing and demonstrating the service performance adaptation mechanisms of the MML

## A.2 EXPERIMENTS WITH REGARD TO ENERGY-RELATED CONSIDERATIONS

The experiments presented in this section have been performed with the goal of evaluating the impact of Web service-specific factors on the energy consumption of Web service-based smartphone applications. The standard factors (device and network type, amount of data) have been controlled as needed, or varied to an extent which is known inside the tested Web service invocation sequences. The next paragraphs explain which two Web-service specific factors have been examined and why.

Device state changes and "device-network tail times" are critical for energy consumption [11, 113]. As these are normally determined by time constants [11], an aspect of Web service invocations that can strongly affect them is the timing of the invocations. Further, the logic of service response caching [104] is an aspect of services computing that can implicitly affect standard energy consumption factors, such as the connection establishments [119] or the amounts of transmitted data [11]. Based on the above, but also on findings from own preliminary experiments, it has been concluded that the following factors may affect the energy consumption of a given sequence of Web service invocations:

- *Temporal distribution of the invocations:* The lengths of the *time intervals between subsequent invocations* can determine how many times the device changes its state[2], but also how long it remains in idle states, which are important for achieving low energy consumption. Generally, it can be assumed that there are three possibilities for the intervals: (i) They are so short that a state change is not probable at all, (ii) they have a length that does not allow certainty about the exact state changes between the calls, and (iii) they are so long that the same state changes should always occur between subsequent invocations. Therefore, it makes sense to talk about three categorical values for these intervals, namely *short*, *medium*, and *long*. A finer granularity is thinkable, but is avoided because it does not seem to be necessary and it would complicate the design of the experiment and the drawing of conclusions.

- *Purpose of Web service response caching:* Because of the device states, but also because of the costs of consumer-provider connection establishments, the purpose with which Web service response caching is applied (or not), may be an important factor. Web service response caching is applied with one of two different goals: Either in order to achieve the transmission of shorter responses, as in [104] and [74], or in order to eliminate transmissions, as in traditional approaches, e.g., [79]. Both approaches have advantages, e.g., the first ones can guarantee response freshness, while the second ones need less connection establishments (cf. Chapter 5). The different handling of connection establishments may lead to big differences with regard to energy consumption. Putting it all together, caching may (i) *not be applied at all*, (ii) *be applied in order to achieve more lightweight messaging*, or (iii) *be applied in order to avoid connection establishments*.

Table 13 summarizes the nine invocation patterns (InvP1-InvP9) that are obtained by combining all possibilities for the two examined factors.

2 The number of possible states, as well as their characteristics, may be device-dependent. More detailed discussions about them can be found in [11] and [179]. However, note that the rest of this section handles them abstractly and without loss of generality.

Table 13: Combining possibilities for the definition of nine Web service invocation patterns

| | | **WS invocation time intervals** | | |
|---|---|---|---|---|
| | | *short* | *medium* | *long* |
| **Use of caching** | *No* | InvP1 | InvP2 | InvP3 |
| | *For lightweight messaging* | InvP4 | InvP5 | InvP6 |
| | *For less connections* | InvP7 | InvP8 | InvP9 |

*Testbed and Setup*

Although parallel validating experiments have been performed on an Android-based Google Nexus One device, as well, all the results presented here come from a Symbian-based Nokia E71 device. This is because the latter offers very detailed energy-profiling possibilities. The same network connections, i.e., the same WLAN and the same UMTS card, have been used for all the experiments.

The software that enabled the measurements consisted of own implementations of Web service-based mobile testbed applications and off-the-shelf energy-profiling applications, which ran in parallel. The latter have been the Nokia Energy Profiler[3] for the Symbian device and the PowerTutor[4] for the Android device. The testbed applications have been implemented with J2ME and the Android SDK, respectively. Special care has been taken so that the Web service invocations have been the only variable energy-consuming activity during the measurements.

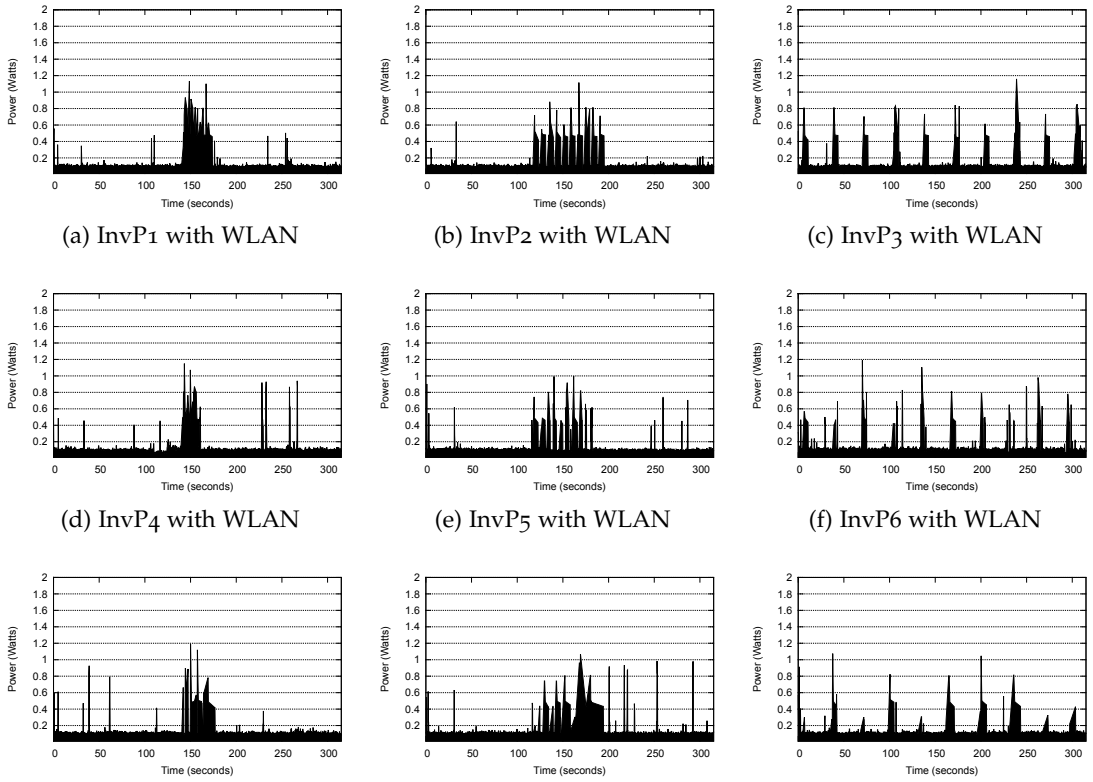
In every single experiment, the same *Web service* has been called the same number of times, namely *ten* times, resulting in responses of the same size, namely ca. 15 kB each time. All single experiments with the same network have had the same total duration (315 seconds for WLAN and 325 seconds for UMTS, because UMTS sometimes needs this extra time to finish all invocations). Only the invocation patterns, as defined in Table 13, have been varied. All energy consumption sources that are not related to the invocation pattern, e.g., display brightness, have been held constant.

In an attempt to give to the *time intervals* (for the given testbed) the meaning they are meant to have, the three categorical values have been interpreted as follows: *short* = 0 seconds (consecutive calls), *medium* = 5 seconds, and *long* = 30 seconds. Experiments with different values could be meaningful. However, the given values reflect the purpose of the definition of the categorical values and have been sufficient for drawing interesting conclusions.

For the invocation patterns that involve caching, a cache hit ratio of 50% has been used. As the approaches imply, in the case of a hit, the responses of InvP7-InvP9 were found on the device, while the responses of InvP4-InvP6 were SOAP messages with trivial body elements (cf. [74, 104]). Keeping the hit ratio constant at 50% has been sufficient for drawing conclusions, as the effects of caching on the amount of transmitted data have been known and taken into account for the analysis.

3 http://www.developer.nokia.com/Resources/Tools_and_downloads/Other/Nokia_Energy_Profiler (Last accessed in January 2012).

4 http://www.powertutor.org (Last accessed in January 2012).




(a) InvP1 with WLAN (b) InvP2 with WLAN (c) InvP3 with WLAN

(d) InvP4 with WLAN (e) InvP5 with WLAN (f) InvP6 with WLAN

(g) InvP7 with WLAN (h) InvP8 with WLAN (i) InvP9 with WLAN

Figure 37: An instance of power consumption over time for each Web service invocation pattern with WLAN

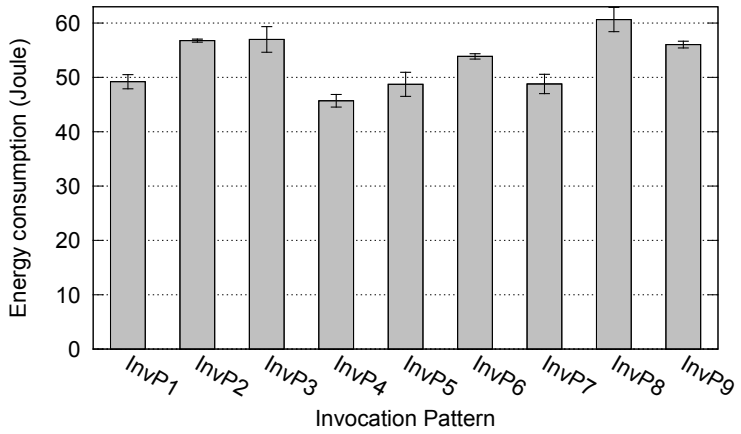
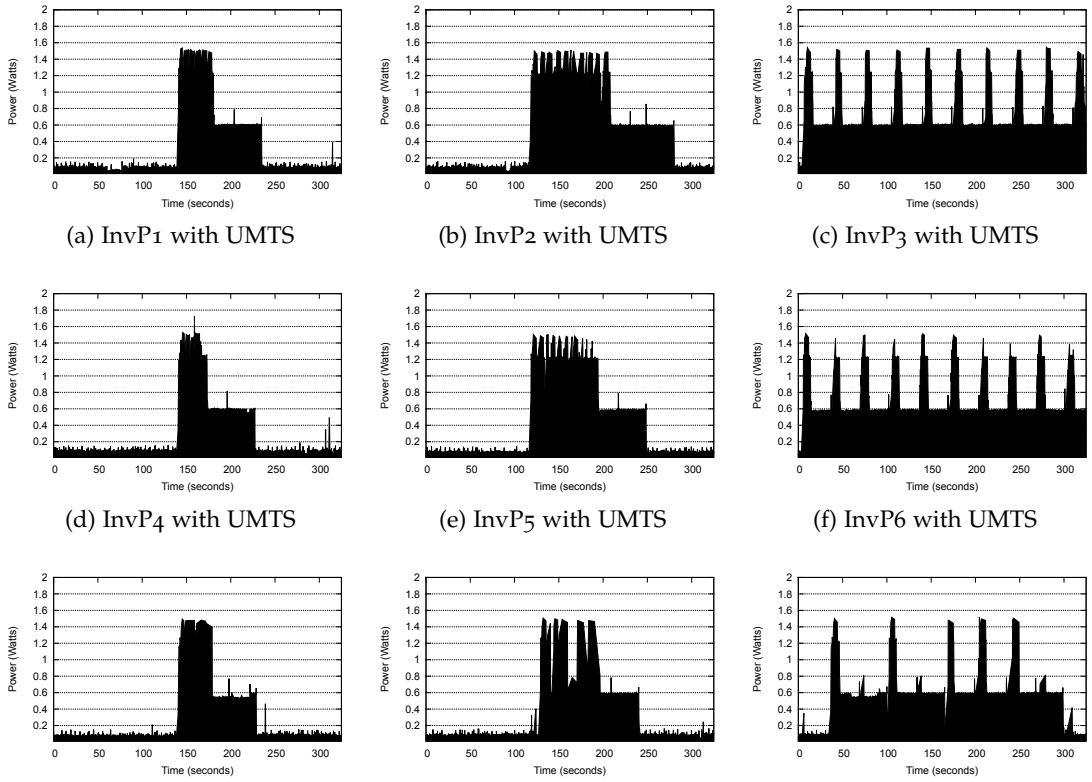



Figure 38: Comparsion of the energy consumption of all Web service invocation patterns for WLAN




(a) InvP1 with UMTS (b) InvP2 with UMTS (c) InvP3 with UMTS

(d) InvP4 with UMTS (e) InvP5 with UMTS (f) InvP6 with UMTS

(g) InvP7 with UMTS (h) InvP8 with UMTS (i) InvP9 with UMTS

Figure 39: An instance of power consumption over time for each Web service invocation pattern with UMTS

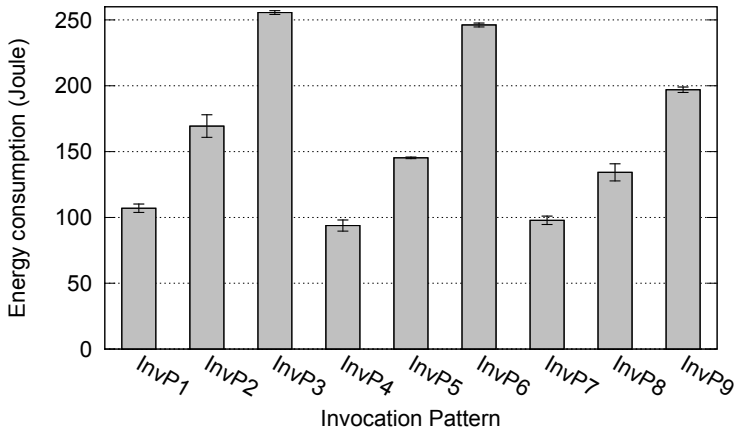



Figure 40: Comparsion of the energy consumption of all Web service invocation patterns for UMTS

*Results and Discussion*

For illustration purposes, only one representative detailed result of "power consumption over time" is presented for each case, i.e., for each combination of an invocation pattern and an access network. However, each case has been repeated three times, with very small deviations in the results. The values of the average consumed energy of each case are presented with 95% confidence intervals in comparative histograms. Thus, Fig. 37 shows the exact power consumption over time for every invocation pattern with a WLAN connection, while Fig. 38 compares the energy consumption[5] of the invocation patterns. Accordingly, Fig. 39 shows the power consumption over time for every invocation pattern with a UMTS connection, while Fig. 40 compares the energy consumption of the invocation patterns.

In the following, first, a summary of the results is provided and, second, three important domain-specific findings are explained.

As can be seen, the Web service invocations dominate completely over any side-operation or background-activity. This is indicated both by the clarity with which the occurrence of the invocations can be "read" on the graphs of Fig. 37 and 39 and by the very small deviations among the repetitions of the experiments (cf. confidence intervals in Fig. 38 and 40). The small peaks[6] that appear independently of the invocations in the case of WLAN connections may appear with a different frequency for different WLAN connections. However, their insignificance in this experiment is shown by the small deviations of the total energy consumption.

In general, the invocation patterns InvP1, InvP4, and InvP7 seem to be the most energy-efficient for both WLAN and UMTS, while InvP5 also presents good results in the case of WLAN. A further remark is that, with some exceptions (e.g.., InvP8 vs. Invp9), the invocation patterns that perform better with a WLAN connection perform better with a UMTS connection, as well. However, the differences are much bigger in UMTS, not only in absolute numbers, but also proportionally. This is a direct consequence of two UMTS characteristics. First, the high power states are higher than those of WLAN although the low ones do not differ. Second, the UMTS tail time "multiplies" the effect of network activity on energy consumption. As a final general remark, it is noted that the results verify the known fact that WLAN is more energy-efficient than UMTS [11, 113], achieving at least 50% (e.g., InvP1, InvP4, InvP7) or even up to 75% (e.g., InvP3) less energy consumption.

A more detailed analysis of the results reveals many differences among the different invocation patterns. Among them, the ones that are most interesting from the point-of-view of services computing and would have not necessarily been expected or foreseen by studying related experiments from other domains, are the following:

- *Clustering of Web service invocations is able to reduce energy consumption:* Although this has been expected for UMTS because of the tail time, the results show how great the effect of the clustering is, achieving from ca. 25% (Inv7 vs. Inv8) up to ca. 60% (Inv1 vs. Inv3) energy reduction. Thus, the handling of subsequent Web service invocations is a very thankful scenario for reducing energy by exploiting UMTS tail times. Interestingly, clustering can often help with WLAN

5 *Energy = power × time*. Thus, energy consumption is calculated as the integral of the curves of power over time, i.e., it is equal to the black-colored areas of the graphs in Fig. 37 and 39

6 Because of router beaconing or similar connection maintenance activity.

connections, as well, although WLAN connections have no tail time and are thought to depend only on the amount of transmitted data [11]. For example, InvP1 consumes ca. 15% less energy than InvP2 and InvP3. This seems to happen due to the higher temporal overlapping of networking and processing activities when the invocations are clustered. Note, for example, that the percentual energy savings achieved by [144] with CPU- and display-related techniques do not exceed the value of 12%.

- *The number of Web service connection establishments is not always a dominating energy consumption factor:* This remark refers to the comparison of caching for less connections with caching for lightweight messaging (e.g., InvP4 vs. InvP7, InvP5 vs. InvP8 etc.). This is a very important finding, because the inability of "caching for lightweight messaging" approaches to reduce connection establishments might be otherwise assumed to be a big disadvantage concerning energy consumption. However, the very small overlapping of networking and processing activities of InvP3, InvP6, and InvP9, as well as the fact that they have bigger intervals between the connection establishments, prevent them from being more energy-efficient than other invocation patterns. Actually, caching has generally not offered great enhancements (cf., for example InvP4 vs. InvP1), though this could be different for bigger response sizes.

- *For Web service invocations, WLAN energy consumption is not proportional to the amount of transmitted data:* Related work has often come to the conclusion that the consumed energy with WLAN connections is proportional to the size of the data transfer [11]. An argument against this has been already provided in the first remark with regard to clustering. Even stronger evidence appears when caching also comes into play. Compare, for example, InvP1-InvP3 with InvP4-InvP6. Although the first ones have transmitted wirelessly almost twice as much data as the second ones, their energy consumption is only slightly higher. This is again because of the characteristics of temporal overlapping of processing and transmission in the scenario of Web service invocations.

## A.3 FURTHER EVALUATION DETAILS WITH REGARD TO DECISION SUPPORT

This Appendix section provides details that have been omitted from the decision support analysis and the corresponding imputation algorithms evaluation of Chapter 6. More concretely, the calculation of the confidence intervals used in the imputation algorithms evaluation is explained. Then, the exact and detailed results of the mean errors of the imputation algorithms, as well as the corresponding standard deviations, are presented in tables.

*Calculation of the confidence intervals for the imputation algorithms evaluation*

For the sake of the representation and the statistical analysis of the evaluation results of Chapter 6, 99% confidence intervals have been used for the mean error in the scoring outputs after the application of the examined imputation algorithms. This practically means that if an infinite number of experiments is performed, i.e., for the complete population of the possible test cases, then the mean errors have a probability of 99% to lie within the calculated confidence intervals. Another way to express this confidence level of 99% is through the so-called significance level, which is denoted as $\alpha$ and is in this case equal to 0.01.

The $100 \times (1-\alpha)\%$ confidence interval of a mean value $e$ is determined by the value

$$i = (z_{1-\alpha/2} \times s)/\sqrt{n}$$

where $z_{1-\alpha/2}$ is the $(1-\alpha/2)$-quantile of a unit normal variate, $s$ is the standard deviation, and $n$ the number of values that have been averaged. The actual interval is then $(e-i, e+i)$. More information about confidence intervals, as well as a table for the calculation of unit normal variate quantiles, can be found in [52].

For the results of Section 6.5, this has meant the following: for a confidence level of 99%, $z_{1-\alpha/2} = 2.576$, while $n$ has been equal to 240, so that $\sqrt{n} = 15.49193338$, and, finally, $i = 0.166280085 \times s$.

*Detailed results of the imputation algorithms evaluation*

Tables 14-19 include the exact values of the mean errors of the imputation algorithms that have been graphically presented in Section 6.5. Additionally, their standard deviations and the corresponding confidence intervals are listed next to them. The results are rounded to ten digits. The meaning and the calculation of the confidence intervals are explained in the next subsection.

Table 14: Exact source data for Figure 24

| IMPUTATION ALG. | AVG. ERROR | STD. DEV. | CONF. INT. |
|---|---|---|---|
| Case Deletion | 1.234 | 1.114 | 0.185 |
| Modus Imputation | 4.853 | 3.546 | 0.590 |
| Random Hot Deck | 0.347 | 0.296 | 0.0493 |
| Dist. Func. Matching | 5.036 | 3.911 | 0.650 |
| Multiple Imputation | 0.387 | 0.313 | 0.052 |
| No Imputation | 12.52 | 4.288 | 0.713 |

Table 15: Exact source data for Figure 25

| IMPUTATION ALG. | AVG. ERROR | STD. DEV. | CONF. INT. |
|---|---|---|---|
| Case Deletion | 0.933 | 0.734 | 0.122 |
| Modus Imputation | 4.856 | 3.383 | 0.562 |
| Random Hot Deck | 0.351 | 0.297 | 0.0494 |
| Dist. Func. Matching | 4.615 | 3.747 | 0.623 |
| Multiple Imputation | 0.373 | 0.318 | 0.0529 |
| No Imputation | 10.36 | 3.097 | 0.515 |

Table 16: Exact source data for Figure 26

| IMPUTATION ALG. | AVG. ERROR | STD. DEV. | CONF. INT. |
|---|---|---|---|
| Case Deletion | 1.143 | 0.956 | 0.159 |
| Modus Imputation | 7.055 | 5.621 | 0.935 |
| Random Hot Deck | 0.420 | 0.348 | 0.0578 |
| Dist. Func. Matching | 7.552 | 4.998 | 0.831 |
| Multiple Imputation | 0.486 | 0.394 | 0.0655 |
| No Imputation | 15.03 | 4.557 | 0.758 |

Table 17: Exact source data for Figure 27

| IMPUTATION ALG. | AVG. ERROR | STD. DEV. | CONF. INT. |
|---|---|---|---|
| Case Deletion | 1.223 | 0.893 | 0.148 |
| Modus Imputation | 4.888 | 3.491 | 0.580 |
| Random Hot Deck | 0.695 | 0.550 | 0.0915 |
| Dist. Func. Matching | 4.688 | 4.168 | 0.693 |
| Multiple Imputation | 0.348 | 0.255 | 0.042 |
| No Imputation | 11.47 | 4.091 | 0.680 |

Table 18: Exact source data for Figure 28

| IMPUTATION ALG. | AVG. ERROR | STD. DEV. | CONF. INT. |
|---|---|---|---|
| Case Deletion | 0.903 | 0.757 | 0.126 |
| Modus Imputation | 4.210 | 3.237 | 0.538 |
| Random Hot Deck | 0.632 | 0.520 | 0.086 |
| Dist. Func. Matching | 4.291 | 3.365 | 0.559 |
| Multiple Imputation | 0.325 | 0.266 | 0.044 |
| No Imputation | 9.501 | 3.008 | 0.500 |

Table 19: Exact source data for Figure 29

| IMPUTATION ALG. | AVG. ERROR | STD. DEV. | CONF. INT. |
|---|---|---|---|
| Case Deletion | 1.442 | 1.230 | 0.204 |
| Modus Imputation | 8.874 | 7.191 | 1.196 |
| Random Hot Deck | 1.234 | 0.902 | 0.150 |
| Dist. Func. Matching | 6.430 | 5.500 | 0.915 |
| Multiple Imputation | 0.378 | 0.303 | 0.050 |
| No Imputation | 13.50 | 4.310 | 0.717 |

# B

# AUTHOR'S PUBLICATIONS

## B.1 MAIN PUBLICATIONS


1. Apostolos Papageorgiou, Marius Schatke, Stefan Schulte, and Ralf Steinmetz. *Lightweight Wireless Web Service Communication through Enhanced Caching Mechanisms*. *International Journal of Web Services Research*, 2012. Invited paper under submission.

2. Apostolos Papageorgiou, Tronje Krop, Sebastian Ahlfeld, Stefan Schulte, Julian Eckert, and Ralf Steinmetz. *Enhancing Availability through Dynamic Monitoring and Management in a Self-Adaptive SOA Platform*. *International Journal on Advances in Software*, 3(3&4):434–446, 2011. ISSN 1942-2628.

3. Apostolos Papageorgiou, André Miede, Dieter Schuller, Stefan Schulte, and Ralf Steinmetz. *Always Best Served: On the behaviour of QoS- and QoE-based Algorithms for Web Service Adaptation*. In: IEEE, editor, *PERCOM Workshops - 8th International Workshop on Managing Ubiquitous Communications and Services (MUCS 2011)*, pages 71–76. IEEE, 2011. ISBN 978-1-61284-936-2.

4. Apostolos Papageorgiou, Marius Schatke, Stefan Schulte, and Ralf Steinmetz. *Enhancing the Caching of Web Service Responses on Wireless Clients*. In: IEEE Computer Society, editor, *9th IEEE International Conference on Web Services (ICWS 2011)*, pages 9–16. IEEE, 2011. ISBN 978-0-7695-4463-2/11.

5. Apostolos Papageorgiou, Jeremias Blendin, André Miede, Julian Eckert, and Ralf Steinmetz. *Study and Comparison of Adaptation Mechanisms for Performance Enhancements of Mobile Web Service Consumption*. In: *Sixth IEEE World Congress on Services (SERVICES 2010)*, pages 667–670. 2010. ISBN 978-0-7695-4129-7/10.

6. Apostolos Papageorgiou, Tronje Krop, Sebastian Ahlfeld, Stefan Schulte, Julian Eckert, and Ralf Steinmetz. *Enhancing Availability with Self-Organization Extensions in a SOA Platform*. In: *Proceedings of the Fifth International Conference on Internet and Web Applications and Services (ICIW 2010)*, pages 161–166. 2010. ISBN 978-0-7695-4022-1.

7. Apostolos Papageorgiou, Bastian Leferink, Julian Eckert, Nicolas Repp, and Ralf Steinmetz. *Bridging the Gaps towards Structured Mobile SOA*. In: *The 7th International Conference on Advances in Mobile Computing & Multimedia (MoMM 2009)*, pages 288–294. ACM, 2009. ISBN 978-1-60558-659-5.

8. Apostolos Papageorgiou, Stefan Schulte, Dieter Schuller, Michael Niemann, Nicolas Repp, and Ralf Steinmetz. *Governance of a Service–Oriented Architecture for Environmental and Public Security*. In: *Information Technologies in Environmental Engineering, ITEE 2009 - Fourth International ICSC Symposium*, pages 39–52. Springer, Berlin Heidelberg New York, 2009. ISBN 978-3-540-88350-0.

## B.2 CO-AUTHORED PUBLICATIONS


9. Sebastian Zöller, Apostolos Papageorgiou, Johannes Schmitt, Marek Meyer, and Ralf Steinmetz. *Innovative Technologie für mobile Fahrgastinformationssysteme*. In: *Proceedings of HEUREKA '11 – Optimierung in Verkehr und Transport*, pages 29–48. Forschungsgesellschaft fuer Strassen– und Verkehrswesen, FGSV Verlag, 2011. ISBN 978-3-941790-72-8.

10. Julian Eckert, Marc Bachhuber, André Miede, Apostolos Papageorgiou, and Ralf Steinmetz. *Readiness and Maturity of Service-oriented Architectures in the German Banking Industry – A Multi-Participant Case Study*. In: *IEEE International Conference on Digital Ecosystems and Technologies 2010 (IEEE DEST 2010)*, pages 270–274. 2010.

11. Dieter Schuller, André Miede, Julian Eckert, Ulrich Lampe, Apostolos Papageorgiou, and Ralf Steinmetz. *QoS-based Optimization of Service Compositions for Complex Workflows*. In: *Proceedings of the Eighth International Conference on Service Oriented Computing (ICSOC 2010)*, pages 641–648. Springer, 2010. ISBN 978-3-642-17357-8.

12. Jeremias Blendin, Liang Han, Apostolos Papageorgiou, André Miede, and Ralf Steinmetz. *An Open-Source Case Study for Service-oriented Architectures with SOPERA*. Technical Report KOM-TR-2009-03, Technische Universität Darmstadt, 2009.

13. Julian Eckert, Deniz Ertogrul, Apostolos Papageorgiou, Nicolas Repp, and Ralf Steinmetz. *The Impact of Service Pricing Models on Service Selection*. In: *4th International Conference on Internet and Web Applications and Services (ICIW'09)*, pages 316–321. 2009. ISBN 978-0-7695-3613-2.

14. Julian Eckert, Tim Lehrig, Apostolos Papageorgiou, Nicolas Repp, and Ralf Steinmetz. *Solving Resource Planning Problems – A Heuristical Solution*. In: *11th International Conference on Information Integration and Web-based Application and Services (iiWAS 2009)*, pages 348–353. ACM, 2009. ISBN 978-3-85403-260-1.

15. André Miede, Jean-Baptiste Behuet, Apostolos Papageorgiou, Michael Niemann, Nicolas Repp, and Ralf Steinmetz. *Qualitative and Quantitative Aspects of Cooperation Mechanisms for Monitoring in Service-oriented Architectures*. In: *Proceedings of the Third IEEE International Conference on Digital Ecosystems and Technologies (IEEE DEST 2009)*, pages 563–568. IEEE, IEEE Computer Society, 2009. ISBN 978-1-4244-2346-0.

16. Dieter Schuller, Apostolos Papageorgiou, Stefan Schulte, Julian Eckert, Nicolas Repp, and Ralf Steinmetz. *Process Reliability in Service–Oriented Architectures*. In: *Proceedings of the Third IEEE International Conference on Digital Ecosystems and Technologies (IEEE DEST 2009), Istanbul*, pages 606–611. IEEE, IEEE Computer Society, 2009. ISBN 978-1-4244-2346-0.

17. Athanasios Antoniou, Ioannis Chatzigiannakis, Athanasios Kinalis, Georgios Mylonas, Sotiris Nikoletseas, and Apostolos Papageorgiou. *A Peer–to–Peer Environment for Monitoring Multiple Wireless Sensor Networks*. In: *International*

*Workshop on Modeling Analysis and Simulation of Wireless and Mobile Systems (MSWiM), 2nd ACM Workshop on Performance Monitoring and Measurement of Heterogeneous Wireless and Wired Networks (PM2HW2N 2007)*, pages 132–135. ACM, 2007. ISBN 978-1-59593-805-3.

# C

## CURRICULUM VITÆ

### PERSONAL INFORMATION

| | |
|---|---|
| Name | Apostolos Papageorgiou |
| Date of Birth | October 4, 1983 |
| Place of Birth | Athens |
| Nationality | Greek |

### EDUCATION

| | |
|---|---|
| 07/2008–today | Technische Universität Darmstadt (Darmstadt, Germany)<br>PhD candidate at the Multimedia Communications Lab (KOM)<br>Department of Electrical Engineering |
| 07/2011 | Yale University (New Haven, USA)<br>Visiting researcher at the Embedded Networks and Applications Lab (ENALAB)<br>Departments of Electrical Engineering and Computer Science |
| 09/2001–11/2007 | University of Patras (Patras, Greece)<br>Dipl.-Ing. at the Computer Engineering and Informatics Department |
| 02/2005–07/2005 | Universidad de Valladolid (Valladolid, Spain)<br>Exchange student at the Telecommunications Engineering Department |
| Until 07/2001 | Greek public high-school (Patras, Greece) |

### PROFESSIONAL EXPERIENCE

| | |
|---|---|
| 07/2008–today | Technische Universität Darmstadt (Darmstadt, Germany)<br>Research assistant at the Multimedia Communications Lab (KOM) |
| 10/2009–today | Hessian Telemedia Technology Competence-center (httc e. V.) (Darmstadt, Germany)<br>Research assistant of the SOA Competence Center (SCC) and leader of the Green Mobility project |
| 05/2007–05/2008 | Greek Army (Greece)<br>Private (spec.: telecommunications technician, compulsory military service) |

| | |
|---|---|
| 03/2003–05/2003 | Information Society Operational Program of the E.U. (Patras, Greece)<br>Informatics assistant teacher at courses for school teachers |

## TEACHING ACTIVITIES

| | |
|---|---|
| 2009, 2012 | Net-centric Systems (NCS) lectures and exercises |
| 2009, 2010 | Lab exercise Multimedia Communications Lab II (supervisor) |
| 2009 | Lab exercise Multimedia Communications Lab II – SOA Integration Project in cooperation with Software AG (co-supervisor) |
| 2009 | Seminar Communication Systems and Multimedia I – Advanced Topics of Future Internet Research (supervisor) |
| 2008–today | Tutor for various Bachelor's and Master's theses (including "Studien-" and "Diplomarbeiten") |

## SCIENTIFIC ACTIVITIES

| | |
|---|---|
| Reviewer | International Conference on Internet and Web Applications and Services (ICIW), 2011, 2012 |
| | International Conference on Information Integration and Web-based Applications & Services (IIWAS), 2009, 2010 |
| Advisory board | Supported the successful application of a startup company (`www.bettertaxi.de`) for an EXIST-grant as a member of the technical advisory board |

## AWARDS AND HONORS

| | |
|---|---|
| 02/2012 | Award "Ausgezeichnetes Konzept 2012" (Excellent concept 2012) of the initiative "Smarter City Karlsruhe", handed over by the Karlsruhe city mayor for the project "Green Mobility" |
| 05/2010 | Best Paper Award at the International Conference on Internet and Web Applications and Services (ICIW 2010) in Barcelona, Spain for the paper: *Enhancing Availability with Self-Organization Extensions in a SOA Platform*, Apostolos Papageorgiou, Tronje Krop, Sebastian Ahlfeld, Stefan Schulte, Julian Eckert, Ralf Steinmetz |

# D

## ERKLÄRUNG LAUT §9 DER PROMOTIONSORDNUNG

Ich versichere hiermit, dass ich die vorliegende Dissertation allein und nur unter Verwendung der angegebenen Literatur verfasst habe.

Die Arbeit hat bisher noch nicht zu Prüfungszwecken gedient.

*Darmstadt, 2012*